DESARROLLO DE UN ALGORITMO DE APRENDIZAJE COMPUTACIONAL PARA EL CONTROL DE MOVIMIENTO DEL ROBOT EMERGE

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Autores
Natalia Rodríguez González
Yeyffer Arbey Montes Fonseca

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
Facultad Tecnológica
Ingeniería en Control y Automatización

Bogotá D.C. Mayo, 2023

DESARROLLO DE UN ALGORITMO DE APRENDIZAJE COMPUTACIONAL PARA EL CONTROL DE MOVIMIENTO DEL ROBOT EMERGE

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Natalia Rodríguez González
Código: 20201383020 - Correo: natarodriguezg@udistrital.edu.co
Yeyffer Arbey Montes Fonseca
Código: 20201383004 - Correo: yeamontesf@udistrital.edu.co

Monografía

Presentado para optar al título de: Ingeniero(a) en Control y Automatización

Director
Luis Fernando Pedraza Martínez
Codirector
Henry Alberto Hernández Martínez

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
Facultad Tecnológica
Ingeniería en Control y Automatización
Bogotá D.C. Mayo, 2023

Dedicatoria

A nuestras familias que nos apoyaron de múltiples formas durante estos años de formación profesional, a nuestros compañeros Andrés Felipe Caro y Arley Felipe Piza por acompañarnos en estos años y a nuestros directores de tesis Luis Fernando Pedraza y Henry Alberto Hernández por su apoyo y confianza en este proyecto.

Índice

Índice de Figuras iv
Índice de Tablas vi
Lista de Abreviaturas y Siglas ix
Resumen x
1. Introducción 11
2. Planteamiento del problema 2
3. Justificación 3
4. Objetivos 4
4.1. Objetivo General 4
4.2. Objetivos específicos 4
5. Marco de referencia 5
5.1. Antecedentes 5
5.1.1. Sistemas de locomoción en robots ápodos 5
5.1.1.1. Mecanismo con ruedas tipo diferencial 5
5.1.1.2. Mecanismo con ruedas tipo síncrona 5
5.1.1.3. Mecanismo con ruedas tipo Ackermann 6
5.1.1.4. Mecanismo bioinspirado 6
5.1.1.5. Mecanismo ápodo 7
5.1.1.6. Robot EMERGE 8
5.1.2. Técnicas de control de movimiento en robots modulares ápodos 9
5.1.2.1. Tablas de control de movimiento 9
5.1.2.2. Cinemática inversa 9
5.1.2.3. Generadores sinusoidales 9
5.1.2.4. Generador central de patrones (CPG) 10
5.1.2.5. Control de movimiento con algoritmos de aprendizaje 10
5.1.3. Aprendizaje computacional en la robótica 11
5.1.3.1. Clasificación 11
5.1.3.2. Redes neuronales artificiales 11
5.1.3.3. Agrupamiento 11
5.1.3.4. Regresión 11
5.2. Marco teórico 12
5.2.1. Robots modulares ápodos 12
5.2.1.1. Morfología tipo cadena 13
5.2.1.2. Morfología tipo retícula 14
5.2.1.3. Morfología tipo hibrida 14
5.2.1.4. Robot modular EMERGE 15

5.2.2. Control de movimiento de robots modulares ápodos 17
5.2.2.1. Generadores sinusoidales 17
5.2.2.2. Cinemática inversa 17
5.2.2.3. Tablas de control de movimiento 17
5.2.2.4. Generador central de patrones (CPG) 18
5.2.3. Inteligencia Artificial 18
5.2.3.1. Machine learning 19
5.2.3.2. Aprendizaje profundo 19
5.2.3.3. Modelos de aprendizaje supervisado 20
5.2.3.4. Modelos de aprendizaje no supervisado 20
5.2.3.5. Modelos de aprendizaje por refuerzo 20
5.2.3.6. TensorFlow-Keras 20
5.3. Marco Legal 21
5.3.1. ISO/IEC AWI TS 8200 21
5.3.2. ISO/IEC TR 24029-1:2021 21
6. Metodología 22
6.1. Control de movimiento del robot EMERGE con tablas de movimiento 22
6.2. Desarrollo algoritmo de control de movimiento con aprendizaje computacional 27
6.3. Implementación algoritmo de control basado en aprendizaje computacional 30
7. Resultados 32
7.1. Estructura robot EMERGE morfología cadena, tipo ELE y bloque 32
7.2. Resultados de entrenamiento del robot con tablas de control de movimiento 35
7.3. Entrenamiento de las redes neuronales 37
7.4. Comparación para el control de movimiento del robot EMERGE con tablas de movimiento y redes neuronales en entornos llanos y con obstáculos 43
8. Conclusiones y Recomendaciones 71
9. Referencias 73
10. Anexos 79
10.1. Anexo 1 79
10.2. Anexo 2 82
10.3. Anexo 3 92
10.4. Anexo 4 99
10.5. Anexo 5 100

Índice de Figuras

Figura 1. Mecanismo con ruedas tipo diferencial Segway. Tomado de (Xataka.com, 2020). 5
Figura 2. Robot tipo síncrono y rueda holonómica. Tomado de (Es.dhgate.com, 2022), (ELECTROSTORE, 2019). 6
Figura 3. Mecanismo con ruedas tipo Ackermann. Tomado de (Aliexpress, 2022). 6
Figura 4. Mecanismos bioinspirado robot hexápodo. Tomado de (Servotronik, 2022). 7
Figura 5. Robot modular. Elaboración propia. 8
Figura 6. Robot EMERGE morfología cadena. Elaboración propia. 9
Figura 7. Algoritmo de control de movimiento por medio de tablas. Tomado de (Morantes et al., 2020). 10
Figura 8. Clasificación y caracterización de los robots modulares. Tomado de (Gonzalez Gomez et al., 2006). 13
Figura 9. Robot ModRED. Tomado de (Baca et al., 2014). 14
Figura 10. Robot tipo retícula Mori. Tomado de (Belke & Paik, 2017). 14
Figura 11. Robot tipo hibrido HexaMob. Tomado de (Reddy et al., 2017). 15
Figura 12. Modulo del robot EMERGE. Elaboración propia. 15
Figura 13. Vista frontal DYNAMIXEL, detalle posición objetivo. Tomado de (ROBOTICS, n.d.) 16
Figura 14. Clasificación de las metodologías de inteligencia artificial (Trappey et al., 2020). 19
Figura 15. Diagrama funcionamiento elaboración tablas de movimiento. Elaboración propia. ... 22
Figura 16. Interfaz gráfica de entrenamiento de tablas de movimiento. Elaboración propia. 23
Figura 17. Robot EMERGE morfologías cadena, bloque y tipo ELE respectivamente, para el entrenamiento de tablas de movimiento. Elaboración propia. 23
Figura 18. Parámetros de trama al controlador de motores Arduino. Elaboración propia. 24
Figura 19. Diagrama de control por tablas de movimiento. Elaboración propia. 24
Figura 20. Diagrama electrónico de conexión del controlador local. Elaboración propia. 25
Figura 21. Pseudocódigo algoritmo de control de movimiento por tablas de movimiento. Elaboración propia. 26
Figura 22. Pseudocódigo del algoritmo implementado en el Arduino. Elaboración propia. 26
Figura 23. Diagrama de la solución implementada para entrenamiento redes neuronales para los modelos de movimiento. Elaboración propia. 27
Figura 24. Algoritmo de entrenamiento de movimientos basado en redes neuronales. Elaboración propia. 28
Figura 25. Diagrama de la solución implementada para el entrenamiento de la red neuronal para el modelo de los sensores. Elaboración propia. 29
Figura 26. Pseudocódigo del algoritmo de control de movimiento por aprendizaje computacional. Elaboración propia. 30
Figura 27. Interfaz gráfica activación redes neuronales. Elaboración propia. 31
Figura 28. Representación de la interfaz gráfica por tipo especifico de morfología. Elaboración propia. 31
Figura 29. Ensamblaje robot EMERGE morfología bloque. Elaboración propia. 32
Figura 30. Configuración morfología bloque. Elaboración propia. 33
Figura 31. Ensamblaje robot EMERGE morfología cadena. Elaboración propia. 33
Figura 32. Configuración morfología cadena. Elaboración propia. 34

Figura 33. Ensamblaje robot EMERGE morfología ELE. Elaboración propia..... 34
Figura 34. Configuración morfología tipo ELE. Elaboración propia..... 35
Figura 35. Gráfica de velocidad actuador (bits) sobre tiempo de envío trama (s) por distancia promedio (cm) en morfología cadena. Elaboración propia. 35
Figura 36. Gráfica de velocidad actuador (bits) sobre tiempo de envío trama (s) por distancia promedio (cm) en morfología ELE. Elaboración propia..... 36
Figura 37. Gráfica de velocidad actuador (bits) sobre tiempo de envío trama (s) por distancia promedio (cm) en morfología bloque. Elaboración propia. 36
Figura 38. Plano de visualización del entorno de pruebas para la morfología cadena con evasión de obstáculo. Elaboración propia. 44
Figura 39. Plano de visualización del entorno de pruebas para la morfología cadena en entorno llano. Elaboración propia..... 44
Figura 40. Representación gráfica morfología cadena experimento con obstáculos. Elaboración propia..... 45
Figura 41. Representación gráfica morfología cadena experimento en entorno llano. Elaboración propia..... 51
Figura 42. Representación gráfica ángulo de desfase morfología cadena experimento en entorno llano. Elaboración propia..... 52
Figura 43. Plano de visualización del entorno de pruebas para la morfología bloque con evasión de obstáculo. Elaboración propia. 53
Figura 44. Plano de visualización del entorno de pruebas para la morfología bloque en entorno llano. Elaboración propia..... 53
Figura 45. Representación gráfica morfología bloque experimento con obstáculos. Elaboración propia..... 55
Figura 46. Representación gráfica morfología bloque experimento en entorno llano. Elaboración propia..... 60
Figura 47. Representación gráfica morfología bloque experimento en entorno llano medición ángulo de desfase. Elaboración propia. 61
Figura 48. Plano de visualización del entorno de pruebas para la morfología tipo ELE con evasión de obstáculo. Elaboración propia..... 62
Figura 49. Plano de visualización del entorno de pruebas para la morfología tipo ELE en entorno llano. Elaboración propia..... 62
Figura 50. Representación gráfica morfología tipo ELE experimento con obstáculos. Elaboración propia..... 63
Figura 51. Representación gráfica morfología tipo ELE experimento en entorno llano. Elaboración propia..... 69
Figura 52. Representación gráfica morfología tipo ELE experimento en entorno llano medición ángulo de desfase. Elaboración propia. 70

Índice de Tablas

Tabla 1. Determinación de la velocidad de comunicación serial entre un controlador y el DYNAMIXEL a partir de la velocidad de transmisión. Tomado de (ROBOTICS, n.d.). 16
Tabla 2. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo cadena sensado de obstáculos en 0,0. Elaboración propia..... 37
Tabla 3. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo cadena sensado de obstáculos en 1, 0. Elaboración propia..... 37
Tabla 4. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo cadena sensado de obstáculos en 0, 1. Elaboración propia..... 38
Tabla 5. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo cadena sensado de obstáculos en 1,1. Elaboración propia..... 38
Tabla 6. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo bloque sensado de obstáculos en 0,0. Elaboración propia..... 39
Tabla 7. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo bloque sensado de obstáculos en 1,0. Elaboración propia..... 39
Tabla 8. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo bloque sensado de obstáculos en 0,1. Elaboración propia..... 40
Tabla 9. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo bloque sensado de obstáculos en 1,1. Elaboración propia..... 40
Tabla 10. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo ELE sensado de obstáculos en 0,0. Elaboración propia. 41
Tabla 11. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo ELE sensado de obstáculos en 1,0. Elaboración propia. 42
Tabla 12. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo ELE sensado de obstáculos en 0,1. Elaboración propia. 42
Tabla 13. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo ELE sensado de obstáculos en 1,1. Elaboración propia. 43
Tabla 14. Resultados experimento morfología cadena con obstáculo a 50cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia. 45
Tabla 15. Resultados experimento morfología cadena con obstáculo a 50cm en la derecha, con control de movimiento por RNA. Elaboración propia. 45
Tabla 16. Resultados experimento morfología cadena con obstáculo a 75cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia. 46
Tabla 17. Resultados experimento morfología cadena con obstáculo a 75cm en la derecha, con control de movimiento por RNA. Elaboración propia. 46
Tabla 18. Resultados experimento morfología cadena con obstáculo a 100cm en la derecha, con control de movimiento por RNA. Elaboración propia. 47
Tabla 19. Resultados experimento morfología cadena con obstáculo a 100cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia. 47
Tabla 20. Resultados experimento morfología cadena con obstáculo a 50cm en la izquierda, con control de movimiento por RNA. Elaboración propia. 48
Tabla 21. Resultados experimento morfología cadena con obstáculo a 50cm en la izquierda, con control de movimiento por tabla de movimiento. Elaboración propia. 48

Tabla 22. Resultados experimento morfología cadena con obstáculo a 75cm en la izquierda, con control de movimiento por RNA. Elaboración propia. 49
Tabla 23. Resultados experimento morfología cadena con obstáculo a 75cm en la izquierda, con control de movimiento por tabla de movimiento. Elaboración propia. 49
Tabla 24. Resultados experimento morfología cadena con obstáculo a 100cm en la izquierda, con control de movimiento por RNA. Elaboración propia. 50
Tabla 25. Resultados experimento morfología cadena con obstáculo a 100cm en la izquierda, con control de movimiento por tabla de movimiento. Elaboración propia. 50
Tabla 26. Resultados experimento morfología cadena en entorno llano, con control de movimiento por tabla de movimiento. Elaboración propia. 51
Tabla 27. Resultados experimento morfología cadena en entorno llano, con control de movimiento por RNA. Elaboración propia. 52
Tabla 28. Resultados experimento morfología bloque con obstáculo a 50cm en la derecha, con control de movimiento por RNA. Elaboración propia. 54
Tabla 29. Resultados experimento morfología bloque con obstáculo a 50cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia. 54
Tabla 30. Resultados experimento morfología bloque con obstáculo a 60cm en la derecha, con control de movimiento por RNA. Elaboración propia. 55
Tabla 31. Resultados experimento morfología bloque con obstáculo a 60cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia. 56
Tabla 32. Resultados experimento morfología bloque con obstáculo a 75cm en la derecha, con control de movimiento por RNA. Elaboración propia. 56
Tabla 33. Resultados experimento morfología bloque con obstáculo a 75cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia. 56
Tabla 34. Resultados experimento morfología bloque con obstáculo a 50cm en la izquierda, con control de movimiento por RNA. Elaboración propia. 57
Tabla 35. Resultados experimento morfología bloque con obstáculo a 50cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia. 57
Tabla 36. Resultados experimento morfología bloque con obstáculo a 60cm en la izquierda, con control de movimiento por RNA. Elaboración propia. 58
Tabla 37. Resultados experimento morfología bloque con obstáculo a 60cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia. 58
Tabla 38. Resultados experimento morfología bloque con obstáculo a 75cm en la izquierda, con control de movimiento por RNA. Elaboración propia. 59
Tabla 39. Resultados experimento morfología bloque con obstáculo a 75cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia. 59
Tabla 40. Resultados experimento morfología bloque en entorno llano, con control de movimiento por RNA. Elaboración propia. 60
Tabla 41. Resultados experimento morfología bloque en entorno llano, con control de movimiento por tablas de movimiento. Elaboración propia. 61
Tabla 42. Resultados experimento morfología tipo ELE con obstáculo a 50cm en la derecha, con control de movimiento por RNA. Elaboración propia. 63
Tabla 43. Resultados experimento morfología tipo ELE con obstáculo a 50cm en la derecha, con control de movimiento por tabla de movimiento. Elaboración propia. 63

Tabla 44. Resultados experimento morfología tipo ELE con obstáculo a 75cm en la derecha, con control de movimiento por RNA. Elaboración propia. 64
Tabla 45. Resultados experimento morfología tipo ELE con obstáculo a 75cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia. 64
Tabla 46. Resultados experimento morfología tipo ELE con obstáculo a 100cm en la derecha, con control de movimiento por RNA. Elaboración propia. 65
Tabla 47. Resultados experimento morfología tipo ELE con obstáculo a 100cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia. 65
Tabla 48. Resultados experimento morfología tipo ELE con obstáculo a 50cm en la izquierda, con control de movimiento por RNA. Elaboración propia. 66
Tabla 49. Resultados experimento morfología tipo ELE con obstáculo a 50cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia. 66
Tabla 50. Resultados experimento morfología tipo ELE con obstáculo a 75cm en la izquierda, con control de movimiento por RNA. Elaboración propia. 66
Tabla 51. Resultados experimento morfología tipo ELE con obstáculo a 75cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia. 67
Tabla 52. Resultados experimento morfología tipo ELE con obstáculo a 100cm en la izquierda, con control de movimiento por RNA. Elaboración propia. 67
Tabla 53. Resultados experimento morfología tipo ELE con obstáculo a 100cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia. 68
Tabla 54. Resultados experimento morfología tipo ELE en entorno llano, con control de movimiento por RNA. Elaboración propia. 69
Tabla 55. Resultados experimento morfología tipo ELE en entorno llano, con control de movimiento por tablas de movimiento. Elaboración propia..... 70

Lista de Abreviaturas y Siglas

Sigla/Abreviatura	Significado
API	Application Programming Interfaces
DL	Deep learning
IA	Inteligencia artificial
CNN	Redes Neuronales Convolucionales
CPG	Generadores Centrales de Patrones
EMERGE	Easy Modular Embodied Robot Generator
ML	Machine learning
RNA	Red Neuronal Artificial
SVM	Máquinas de Vector de Soporte

Resumen

Esta tesis presenta un algoritmo de control de movimiento del robot EMERGE basado en aprendizaje computacional, para ello se desarrolló un algoritmo de control de movimiento implementando aprendizaje profundo usando la librería TensorFlow – Keras en el entrenamiento de las redes neuronales para las morfologías de cadena, tipo ELE y bloque o cuadropodo; el segundo control de movimiento aplicado para la comparación de la técnica desarrollada fueron las tablas de movimiento siguiendo la técnica de (Morantes et al., 2020) aplicándola para las tres morfologías anteriormente mencionadas. Para elaborar el controlador local del robot, se implementó una Raspberry PI 3 junto con un Arduino Uno, dos sensores ultrasónicos y dos reguladores de voltaje, uno para la alimentación de la Raspberry y el otro para la alimentación de los sensores y el Arduino. Con este trabajo, se incorpora una nueva técnica de control de movimiento en robots modulares aplicando la inteligencia artificial y abriendo el campo de estudio en esta técnica de control para estos robots en las morfologías cadena, tipo ELE y bloque, así como mejorar los tiempos de desplazamiento y el ángulo de desfase con respecto al origen en el eje Y en las tres morfologías con respecto a la técnica de comparación; también se valida la aplicación de redes neuronales de tipo multicapa en el control de movimiento de robots modulares ápodos.

Palabras claves: Control de movimiento, EMERGE, redes neuronales, tablas de movimiento.

1. Introducción

En la actualidad, se almacenan grandes cantidades de datos que han llevado al nacimiento de tecnologías como la inteligencia artificial (IA) para el uso diverso de estos datos, como lo son el desarrollo de chatbots, buscadores inteligentes como ChatGPT, automóviles autónomos como los de la marca Tesla, tecnología aplicada al hogar como Alexa y Google home (Oracle Colombia, n.d.). Los datos almacenados permite que por medio de diversas técnicas de aprendizaje computacional que imitan el aprendizaje humano, se optimice la curva de aprendizaje de estos algoritmos a una mayor velocidad como lo son las técnicas que abarca el machine learning (ML), permitiendo que hoy en día puedan predecir comportamientos en diferentes áreas, una de ellas es el control de movimiento de los robots ápodos (Amazon Web Services (AWS), n.d.).

Sin embargo, en la actualidad el control de movimiento en robots modulares ápodos no es conciso, por ello, en el presente documento se implementó un algoritmo de control de movimiento basado en inteligencia artificial para el robot modular EMERGE, con el objetivo de proporcionar un sistema de control centralizado que se basa en aprendizaje computacional que le permita desplazarse en entornos llanos y con obstáculos.

Para esto, se implementó un algoritmo de control basado en la técnica de aprendizaje profundo en el robot modular EMERGE en tres morfologías diferentes, por medio del lenguaje de programación Python con la librería Tensorflow - Keras en un controlador centralizado, junto sensores ultrasónicos para la detección de obstáculos, cuyo desarrollo se expondrá con más detalle a lo largo de este documento.

Por lo anterior, este trabajo se divide en 7 secciones. La primera sección contempla una introducción al proyecto desarrollado; la segunda sección se compone del planteamiento del problema, allí se describe por qué es importante el desarrollo e implementación de este algoritmo de control en el robot EMERGE; en la tercera sección está la justificación del desarrollo de este proyecto, las limitaciones actuales y los beneficios del desarrollo del mismo; en la cuarta sección se presentan los objetivos a alcanzar de este proyecto; en la quinta sección se presentan los antecedentes de los robots modulares ápodos, las técnicas de control de movimiento y el aprendizaje computacional aplicado al control de movimiento de robots modulares, así como conceptos teóricos y legales sobre el control de movimiento con IA; en la sexta sección se describe detalladamente los procedimientos realizados para la elaboración e implementación del algoritmo de control de movimiento del robot EMERGE basado en inteligencia artificial. Por último, en la séptima sección se explican los resultados obtenidos con el algoritmo y se presenta la respectiva comparación con un algoritmo de control de movimiento basado en tablas de movimiento.

2. Planteamiento del problema

En la actualidad, el aprendizaje computacional o machine learning es una de las ramas que compone la inteligencia artificial, pues esta es aplicable a diferentes áreas del conocimiento como lo son las ciencias exactas, las ingenierías, ciencias sociales, entre otras (FBI, n.d.), (Matich, 2001). Esta característica se aprovecha en múltiples disciplinas como lo es la robótica, la cual se fundamenta en el apoyo de tareas difíciles o riesgosas para la integridad del ser humano. Por ejemplo, en el ensamble de automóviles o el rescate de personas en zonas de difícil acceso (Matich, 2001), (Filio García et al., 2019), (Rico et al., 2013).

Hoy en día se encuentran diferentes tipos de locomoción en robots móviles, entre ellos los robots modulares los cuales con el avance tecnológico han evolucionado en las estrategias de control de movimiento con la finalidad de que cada vez sean más autónomos, no obstante, estas técnicas de control de movimiento de robots modulares aún son materia de estudio (Rico et al., 2013). Actualmente no hay un estudio conciso del control de movimiento de los robots modulares y por esto se propone la implementación de un algoritmo de aprendizaje computacional para el control de movimiento del robot modular EMERGE, el cual permita desarrollar una nueva técnica de control de movimiento de este tipo de robots.

3. Justificación

Las estrategias de control de movimiento de robots móviles representan un desafío en el avance de la robótica, pues es una disciplina que es necesaria en distintas aplicaciones como en el monitoreo, rescate, exploración o para facilitar tareas que requieran de gran precisión como en el ensamble de vehículos o de fuerza desmesurada como en la fabricación de equipos para la generación de energía.

El aprendizaje computacional en el control de movimiento de un robot ápodo, como lo es el EMERGE, implicaría una potencialización de las características del robot como el desplazamiento en superficies o entornos irregulares, ya que, estos mejorarían su capacidad de movimiento autónomo, así como un avance tecnológico para el país. Además, en la universidad sería una herramienta para motivar la implementación de este tipo de algoritmos de control de movimiento en otros sistemas en las distintas líneas de investigación de los grupos de investigación como DIGITI, INTEGRA, XUE, ROMA, entre otros, que podrían desarrollar nuevas aplicaciones con estos algoritmos generando nuevas soluciones. Por esta razón, se pretende desarrollar e implementar un algoritmo de aprendizaje computacional para el control de movimiento del robot modular EMERGE, de esta manera motivando a la investigación e implementación de este algoritmo en aplicaciones de distintas áreas del conocimiento.

4. Objetivos

4.1. Objetivo General

Desarrollar un algoritmo de aprendizaje computacional que genere patrones de movimiento de forma local que le permita al robot EMERGE moverse en entornos llanos y con obstáculos.

4.2. Objetivos específicos

- Desarrollar un algoritmo de aprendizaje computacional que genere patrones de movimiento para tres morfologías diferentes del robot EMERGE.

- Implementar el algoritmo desarrollado en el robot y verificar su funcionamiento en entornos llanos y con obstáculos.

- Comparar los parámetros del robot con el algoritmo propuesto contra la técnica desarrollada por otro autor.

5. Marco de referencia

Esta sección se compone del análisis del estado del arte en el control de movimiento en robots modulares, así como de la aplicación del aprendizaje computacional en robots. También se incluyen las definiciones y el marco teórico necesario que sustentan la presente investigación. Por último, se presenta la normatividad aplicable al presente desarrollo.

5.1. Antecedentes

En esta sección, se presenta un análisis del estado del arte de los sistemas de locomoción en robots ápodos, así como de los distintos mecanismos existentes. Seguidamente, se estudian las técnicas de control de movimiento en robots ápodos, y finalmente se analizan las diferentes aplicaciones de aprendizaje computacional en robótica.

5.1.1. Sistemas de locomoción en robots ápodos

En la robótica móvil la locomoción permite ejecutar tareas especialmente dedicadas a la navegación y movimiento, a través de patrones de movimientos rítmicos hace que el dispositivo genere un recorrido de un punto X a un punto Y. Existen diferentes mecanismos que permiten el movimiento de un robot. El mecanismo generalmente usado es el de las ruedas el cual tiene diferentes variaciones, también existen los de tipo bioinspirado y los ápodos.

5.1.1.1. Mecanismo con ruedas tipo diferencial

Es un sistema de locomoción estructurado por dos ruedas fijas situadas en un eje común del robot, la estructura es mecánicamente sencilla, sin embargo, la implementación y el control del robot son complejos porque la estructura no es estable, por esto el robot cuenta con un sistema de control capaz de hacer que la estructura del robot encuentre el equilibrio, como los estudiados por (Iporre Rivas, 2014) y (López García, 2011). Este tipo de mecanismos son implementados en los segways (Segway, n.d.) que son robots móviles manipulados por personas, como se ve en la Figura 1, para hacer recorridos en cortas distancias.

Figura 1. Mecanismo con ruedas tipo diferencial Segway. Tomado de (Xataka.com, 2020).

5.1.1.2. Mecanismo con ruedas tipo síncrona

Es un mecanismo que cuenta usualmente con tres ruedas fijas, donde cada una de ellas proporciona fuerza de tracción al robot y direccionamiento controlado de la orientación, estos robots suelen

implementar en su estructura ruedas omnidireccionales como se ve en la Figura 2 y con el fin de modificar la dirección instantáneamente sin necesidad de rotar, estos robots son llamados holonómicos y han sido implementados por (Filio García et al., 2019) y (S. Martínez & Sisto, 2009).

Figura 2. Robot tipo síncrono y rueda holonómica. Tomado de (Es.dhgate.com, 2022), (ELECTROSTORE, 2019).

5.1.1.3. Mecanismo con ruedas tipo Ackermann

Es un mecanismo caracterizado por tener cuatro ruedas como se ve en la Figura 3, las delanteras que son las encargadas de dar dirección al robot y las traseras que son las encargadas del desplazamiento del robot. Es un mecanismo que se emplea en los automóviles, donde la dirección la dan las llantas o eje delantero, este tipo de sistemas mecánicos son ampliamente utilizados en la vida cotidiana y han sido estudiados por (ELECTROSTORE, 2019), (Marín et al., n.d.), (Adamu et al., 2018) y (Hrbáček et al., 2010). Sin embargo, es un mecanismo propenso a volcamientos en superficies no llanas.

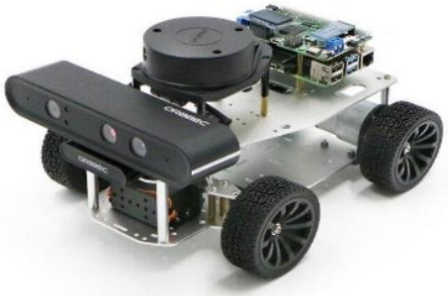


Figura 3. Mecanismo con ruedas tipo Ackermann. Tomado de (Aliexpress, 2022).

5.1.1.4. Mecanismo bioinspirado

Los mecanismos bioinspirado cuentan con un movimiento definido por la cantidad de patas, pueden tener dos extremidades que son llamados bípedos como los estudiados e implementados por (Aguilar & Vivas, 2009), (Magdalena Layos, 1994) y (Toapanta Paucar, 2019) para la detección de obstáculos, así como para el diseño de futuras prótesis; también existen mecanismos de cuatro patas o cuadrúpedos estudiados e implementados por (Calle et al., 2007), (Robles Atuesta, 2020) y

(Mortes Esquer, 2021) para el estudio de técnicas de locomoción en máquinas caminantes, de igual manera para el desarrollo de plataformas robóticas abiertas de bajo costo al igual que para imitar algunos los movimientos característicos de animales de cuatro patas. Por último, están los de seis patas o hexápodos, que se pueden ver en la Figura 4, estudiados por (Rodríguez-Calderón et al., 2015) y (Pérez Machorro, 2009) para determinar las características de movimiento, estabilidad y desplazamiento, tal como para encontrar nuevas técnicas de control de movimiento por medio de la implementación de redes neuronales para la creación de robots más autónomos. Estos mecanismos surgen como una alternativa de locomoción a los sistemas implementados con ruedas gracias a su capacidad de movimiento en terrenos inestables o de difícil acceso, el control de movimiento de estos robots puede ser más complejo con respecto a los de ruedas.

Figura 4. Mecanismos bioinspirado robot hexápodo. Tomado de (Servotronik, 2022).

5.1.1.5. Mecanismo ápodo

A partir de la necesidad de poder tener robots que tuvieran una mejor adaptación al entorno, se realizaron nuevas investigaciones y diseños mecánicos de estructuras más versátiles, flexibles y con mejor estabilidad, surgen los robots ápodos o sin patas como lo evidencian (Méndez Polanco et al., 2006) y (Ellips Masehian, 2015) y con estos los robots modulares (ver la Figura 5) que han inspirado a autores como (Belke & Paik, 2017), (Shigemune et al., 2016) y (Lee & Sanderson, 2002) para el diseño y elaboración de robot modulares inspirados en el origami, así como a la investigación del modelo geométrico de estos como lo han hecho (Aurelian, 2015), tal como (Ahmadzadeh et al., 2016) y (Alberto Brunete et al., 2017) en el estudio de las características y posibles aplicaciones de este tipo de robots. El movimiento de estos robots depende directamente de su estructura mecánica, son bio-inspirados generalmente en animales cuyo movimiento es similar a patrones de movimiento sinusoidal, giratorio o en espiral, como lo desarrolla (Hernández Martinez, 2018) mediante la estrategia de adaptación implementando un generador central de patrones o CPG.

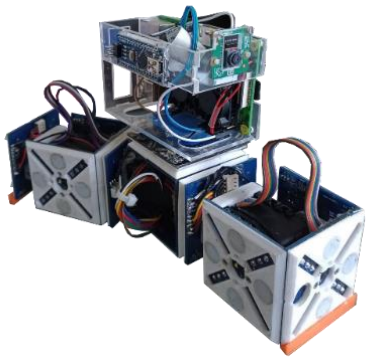


Figura 5. Robot modular. Elaboración propia.

5.1.1.6. Robot EMERGE

El robot modular EMERGE (Easy Modular Embodied Robot Generation) se compone de una serie de módulos que permiten la construcción rápida de morfologías unidimensionales, bidimensionales y otras formas más complejas como se ve en la Figura 6. Esta configuración modular proporciona al robot adaptabilidad a su entorno, lo que permite llevar a cabo experimentos de locomoción y evolución, tal como lo presentan (Moreno et al., 2017).

Autores como (Arenas Morantes et al., 2020) han implementado técnicas de simulación de patrones de movimiento en configuraciones tipo cadena de una dimensión, incluyendo el estudio de parámetros para evitar el desgaste de la estructura mecánica durante la realización de múltiples experimentos. Por otra parte, se han elaborado propuestas como la de (Gómez et al., 2020) en la cual se elaboró un sistema de comunicación que permitió enviar patrones de movimiento al controlador local del robot por medio de tecnología Wi-Fi basada en radio definida por software (SDR) lo cual disminuyó la susceptibilidad del sistema de comunicación al ruido y permitió a través de esta comunicación alojar datos de desempeño del robot en línea con el servidor ThingSpeak de MathWorks.

De la misma forma, los autores (Hernández Martínez et al., 2020) realizaron una evaluación del funcionamiento del aprendizaje de una red neuronal de aprendizaje profundo en la elaboración de movimientos para la morfología tipo cadena a partir del aprendizaje de la red, de los movimientos obtenidos de un algoritmo basado en un generador central de patrones (CPG). Por último, los autores (Conde González & Mahecha Mora, 2021) implementaron un esquema de control de movimiento del robot basado en la detección de obstáculos por medio de redes neuronales que permiten la detección de figuras tridimensionales sin importar la vista de estas dado la robustez de la base de datos con la cual se entrenó la red neuronal.

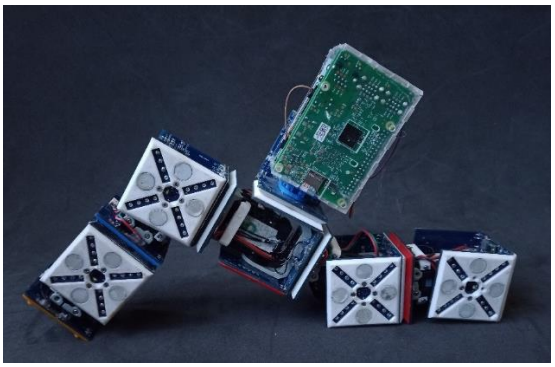


Figura 6. Robot EMERGE morfología cadena. Elaboración propia.

5.1.2. Técnicas de control de movimiento en robots modulares ápodos

La implementación de morfologías complejas o acoplar una gran cantidad de módulos en el robot hace que la actividad del robot se vea afectada en la generación patrones de locomoción continuos, estables y eficientes. Por medio de las técnicas de control de movimiento es posible que el robot pueda ejecutar sus tareas de navegación y recorrido establecido, dentro de las más conocidas se encuentran las tablas de control de movimientos, cinemática inversa implementada por (Felipe et al., 2006), generadores sinusoidales, generador central de patrones (CPG) estudiada e implementada por (Hernández Martinez, 2018) y (Zamorano Martín, 2011).

5.1.2.1. Tablas de control de movimiento

Esta técnica consiste en que todos los parámetros de la rutina de movimiento se encuentran registrados jerárquicamente en un vector que alberga las respectivas posiciones de cada articulación. Esta técnica es implementada para la locomoción del robot modular tipo cadena EMERGE de (Morantes et al., 2020), así mismo (Yim, 1994) estudia el control de movimiento de un robot modular reconfigurable por medio de la técnica descrita anteriormente, los autores (A Brunete et al., 2006) implementan este control de movimiento para un brazo robótico de diferentes modalidades de ejecución.

5.1.2.2. Cinemática inversa

Esta técnica está basada en el uso de ecuaciones cinemáticas para el control de movimiento de cada módulo, esta técnica es implementada por (Vele G, 2005) para el control de un robot bípedo, mediante resolución analítica simplifican el modelo de problema cinemático inverso PCI, (Ibarra Peñaranda et al., 2021) incorporan este método basado en un interpolador cubico de splines con la finalidad de desarrollar un robot capaz de imitar y aprender movimientos de las extremidades superiores del humano; (Ibarra-Peñaranda et al., 2021) desarrollan un aprendizaje de trayectorias con ayuda de un Kinect para obtener las coordenadas espaciales del cada módulo para aplicarlas a un modelo matemático para determinar las posiciones articulares y así calcular el modelo cinemático del robot para desarrollar una rutina de control que establezca la relación entre las velocidades de las articulaciones.

5.1.2.3. Generadores sinusoidales

Este método de control se basa en realizar una ecuación sinusoidal la cual entrega el valor de la posición del módulo, a partir de la estimación de la posición de los demás módulos se genera un

desfase respectivo a cada uno de estos generando el movimiento sinusoidal. Los autores (Morantes et al., 2020) implementan y simulan este método en el robot modular tipo cadena EMERGE donde se le asigna el posicionamiento de cada módulo acoplado al robot con la ecuación característica de este tipo de control, el algoritmo implementado se puede ver en Figura 7.

```
Selección de Entorno
Iniciar variables
Iniciar puerto serie
Tiempo = 0
Valores_S = [vector de posición módulos para el robot real]
Valor función seno = [valores módulos realidad virtual]
Función Seno ()
Dependiendo de la selección de entorno se genera la función seno ()
        Valor_módulo1=Sen (x+ ϑ1)
        Valor_módulo2=Sen (x+ ϑ2)
        Valor_módulo3=Sen (x+ ϑ3)
            Valor_módulo4=Sen (x+ ϑ4)
        Valor_módulo5=Sen (x+ ϑ5)
        Llenar vector [Valor función seno]
            Exportar valor función seno ()
Función entorno ()
Cargar entorno virtual
Cargar valor función seno
   Mientras tiempo < 5000 haga
Enviar valores de realidad virtual [valor de la función seno]
Conversión de valores realidad virtual a módulos =
(1024*Valor_módulo1,2,3,4,5)/π
Enviar valores realidad virtual al módulo LoRa por puerto serial [Valores_S]
Ejecutar la rutina de movimiento durante 500 milisegundos
Tiempo ++
   Fin Mientras
```

Figura 7. Algoritmo de control de movimiento por medio de tablas. Tomado de (Morantes et al., 2020).

5.1.2.4. Generador central de patrones (CPG)

Este método de control está basado en redes neuronales, consiste en replicar movimientos de animales a partir de patrones de movimiento coordinados (Pouya et al., 2011). En (Hernández Martinez, 2018) es implementada esta técnica en el robot modular EMERGE para la simulación del movimiento de una serpiente, así como también es usada por (Zamorano Martín, 2011). Los autores (Moreno García, 2010) implementan este método para desarrollar un modelo de control hibrido para incrementar la autonomía en la generación de movimientos del robot, usando el CPG para proveer al sistema de generación de primitivas de movimiento.

5.1.2.5. Control de movimiento con algoritmos de aprendizaje

Este método de control está basado en diferentes técnicas de inteligencia artificial como lo es el algoritmo de regresión de proyección ponderada localmente (LWPR) implementada por (Ojeda et al., 2017) para la optimización del aprendizaje del modelo interno de un solo robot modular para ordenar al robot a seguir una trayectoria deseada con el respectivo efecto final, así como redes neuronales de aprendizaje profundo usada por (H. H. Martínez et al., 2020) para el control de movimiento de un robot modular tipo cadena.

5.1.3. Aprendizaje computacional en la robótica

El aprendizaje computacional o Machine *Learning* es una rama de los algoritmos computacionales la cual se creó para simular o asemejar la inteligencia humana mediante el aprendizaje de eventos pasados, con la finalidad de inducir modelos sin la intervención o asistencia humana (Reyes García et al., 2018). Esta rama ha sido desarrollada a un ritmo acelerado en siglo 21, (Maglogiannis et al., 2021) estima que 800 trillones de dólares han sido invertidos por parte de la industria automotriz para el diseño y desarrollo de vehículos autónomos capaces de conducirse por sí mismos, como los fabricados por la empresa TESLA.

Dentro de los algoritmos de aprendizaje computacional se encuentran los algoritmos por clasificación, redes neuronales, agrupamiento y regresión.

5.1.3.1. Clasificación

Los algoritmos de clasificación cumplen la función de inducir automáticamente un modelo que a partir de los parámetros con los que se le programe de como respuesta una etiqueta, para esto es necesario suministrarle datos de aprendizaje (Reyes García et al., 2018). Esta técnica es estudiada por (Behal et al., 2009) para el control de movimiento de un robot bípedo, así como (Kuwayama et al., n.d.) implementa un algoritmo de clasificación tipo árbol para la generación de movimiento de un robot humanoide.

5.1.3.2. Redes neuronales artificiales

Las redes neuronales son algoritmos de tratamiento de información, están inspirados o basados en las neuronas del sistema nervioso humano, su funcionamiento se basa en el aprendizaje constante por medio de eventos (Matich, 2001). Hoy en día son algoritmos ampliamente utilizados, como en el control de péndulos invertidos implementado por (Anderson, 1989) para resolver problemas de evaluación de funcionamiento y aprendizaje de funciones no lineales; (Bhattacharya et al., 2018) implementa un algoritmo de redes neuronales para el control de la marcha de un robot caminante; (Behal et al., 2009) estudia el uso de diferentes algoritmos de aprendizaje computacional, entre ellos las redes neuronales, para el control de un robot bípedo.

5.1.3.3. Agrupamiento

Estos algoritmos tienen la funcionalidad de agrupar o segmentar datos automáticamente, son algoritmos de aprendizaje no supervisados, funcionan mediante el agrupamiento de datos en conjuntos con la finalidad de que los datos que están dentro del grupo sean muy parecidos entre sí pero no se parezcan con los datos de otros conjuntos (Reyes García et al., 2018). (Behal et al., 2009) estudia el uso de este tipo de algoritmos para el control de movimiento de un robot bípedo inspirado en las extremidades bajas de un humano.

5.1.3.4. Regresión

Los algoritmos de regresión se fundamentan en la inducción automática de modelos a partir de los datos suministrados previamente con la finalidad de dar como respuesta un valor continuo (Alpaydin, 2014). (Behal et al., 2009) estudia el algoritmo de máquinas de vectores de soporte (SVM) con base en el funcionamiento de los algoritmos de regresión.

5.2. Marco teórico

En esta sección, se presentan las definiciones, conceptos y aspectos teóricos necesarios para el desarrollo de esta investigación. Estos se dividen en tres subsecciones correspondientes a los robots modulares ápodos en el numeral 5.2.1, el control de movimiento de robots modulares ápodos en el numeral 5.2.2, por último, la inteligencia artificial en el numeral 5.2.3.

5.2.1. Robots modulares ápodos

Los robots modulares están conformados por dos o más piezas/módulos que le permiten acoplarse entre sí para formar diferentes morfologías ya sean unidimensionales, bidimensionales o tridimensionales, permitiendo que se adapte al entorno gracias a su diversidad de morfologías (González Gomez, 2008), (Yim, 1994), (Lee & Sanderson, 2002). Debido a que los módulos deben estar en constante sincronización se implementan diferentes protocolos de comunicación para enviar y recibir información de cada módulo ensamblado; estos datos pueden ser, posición y estado de cada actuador o datos referentes a sensores incorporados, con la finalidad de que haya un desplazamiento coordinado y colaborativo entre los módulos (Ye et al., 2018), (Marbach & Ijspeert, 2004), (Rodríguez Lombana, 2019), (Alattas et al., 2019).

El grado de libertad de estos robots está ligado directamente al diseño mecánico de cada módulo, así como de los actuadores integrados los cuales se encargan de generar movimiento, prismático o rotacional. De esta manera, el grado de libertad está determinado por el la cantidad de módulos acoplados y el tipo de morfología implementada (Yim, 1994), (Rodríguez Lombana, 2019), (Chen & Yim, 2008).

Las caras de cada módulo son las encargadas del acoplamiento y/o ensamble con los distintos módulos, el diseño de estas se basa en el tipo de combinaciones realizables y en la morfología a implementar, estas estructuras suelen ser de forma geométrica básica y en algunos casos más elaboradas dado la morfología que se implemente (A Brunete et al., 2006), (Alattas et al., 2019). Existen tres categorías generales que abarcan los diferentes tipos de morfologías de robots modulares, una clasificación más detallada y precisa de este tipo de robots se puede ver en la Figura 8.

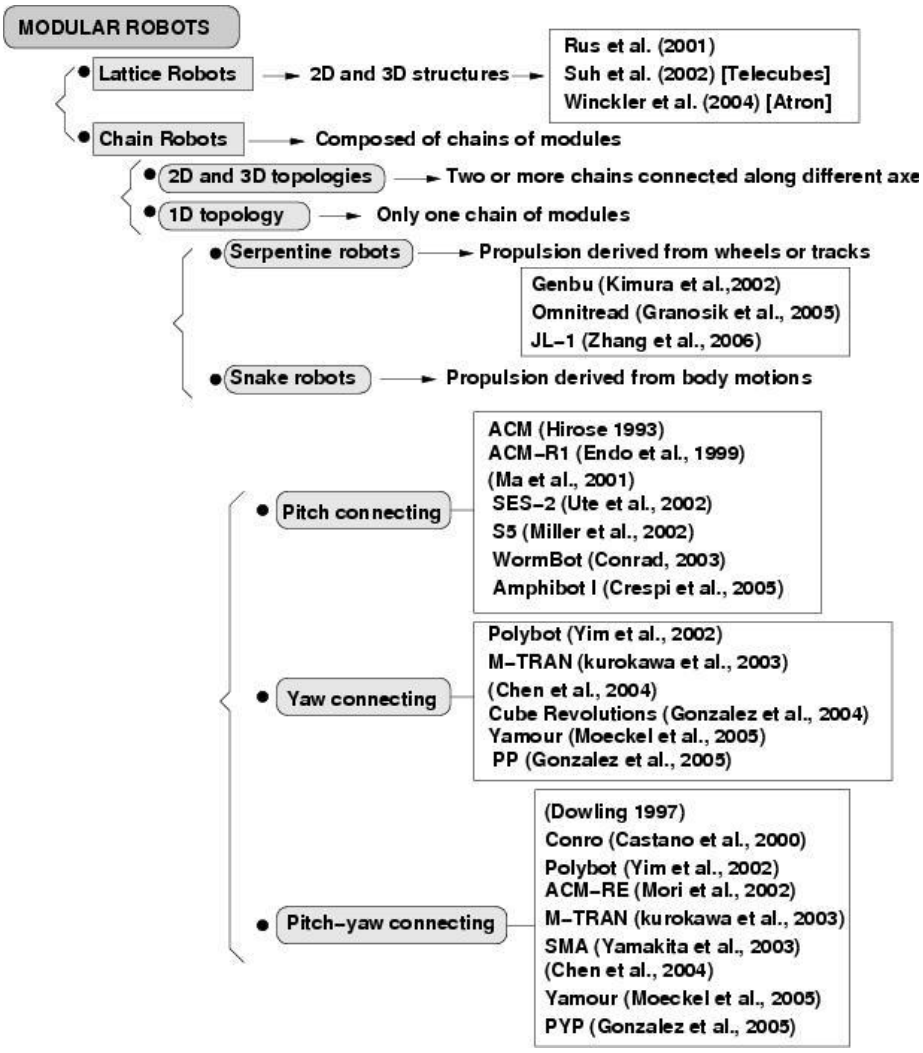


Figura 8. Clasificación y caracterización de los robots modulares. Tomado de (Gonzalez Gomez et al., 2006).

5.2.1.1. Morfología tipo cadena

Esta morfología se caracteriza porque los módulos se conectan en serie simulando la estructura de extremidades, estos se pueden unir entre sí en estructuras arbitrarias, complejas y asimétricas, son estructuras con pocos grados de libertad, pero son flexibles (Miño Rubiano, 2017). Normalmente, es una morfología difícil de representar, controlar y coordinar (Ahmadzadeh et al., 2016)(Moreno García, 2010)(Alattas et al., 2019). Uno de los robots más conocidos de este tipo es el ModRED, el cual en cada uno de sus módulos cuenta con cuatro grados de libertad, de estos tres son rotacionales y uno prismático, permitiendo facilidad en la reconfiguración y locomoción del robot como se ve en la Figura 9 (Baca et al., 2014). Algunos otros robots modulares con esta estructura son M-TRAN, Polybot, CONRO (Marbach & Ijspeert, 2004).

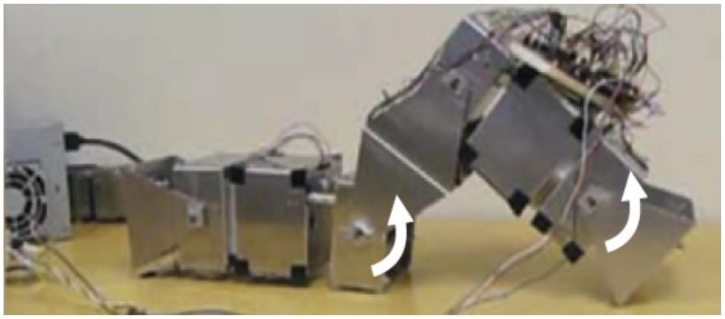


Figura 9. Robot ModRED. Tomado de (Baca et al., 2014).

5.2.1.2. Morfología tipo retícula

Esta estructura está basada en el acoplamiento de módulos entre sí para la formación de figuras bidimensionales o tridimensionales, los módulos de esta morfología no pueden moverse independientemente. Este tipo de morfologías son implementadas ampliamente en el modelamiento y solución de problemas de planificación de la reconfiguración puesto que estos módulos pueden ser controlados en paralelo (Ahmadzadeh et al., 2016)(Alattas et al., 2019)(Miño Rubiano, 2017). Uno de los robots más conocidos de este tipo es el robot Mori, inspirado en el arte japones del origami, sus módulos se ensamblan de manera sencilla por su geometría simétrica y básica como se ve en la Figura 10 (Belke & Paik, 2017). Otros robots tipo retícula desarrollados son el Telecube, Crystalline, I-Cubes, Fractum y ATRON (Marbach & Ijspeert, 2004).

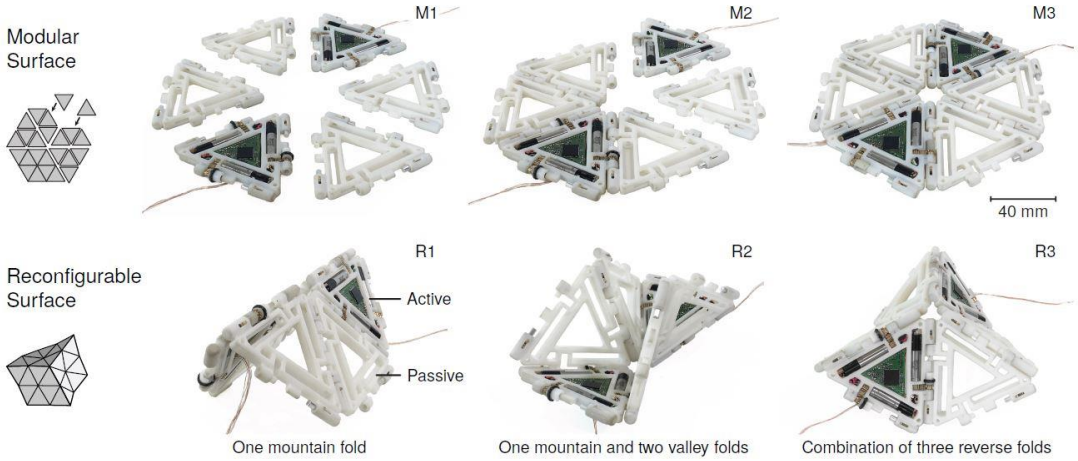


Figura 10. Robot tipo retícula Mori. Tomado de (Belke & Paik, 2017).

5.2.1.3. Morfología tipo hibrida

En esta morfología se fusionan las estructuras anteriormente mencionadas, los módulos de estos robots se interconectan como en las estructuras tipo cadena, pueden tener la misma locomoción que los tipo cadena pero pueden formarse en estructuras o figuras reticulares (Ahmadzadeh et al., 2016). El HexaMob es uno de los robots con este tipo de morfologías, su estructura permite ensamblar figuras dimensionales, bidimensionales o tridimensionales, cada uno de sus módulos está conformado por un chasis frontal, uno trasero y una unidad móvil como se ve en Figura 11; este robot puede reconocer y moverse dependiendo de los obstáculos del entorno (Reddy et al.,

2017). De esta morfología se han desarrollado otros robots como el M-TRAN I, M-TRAN II, Superbot, M-TRAN III, Molecubes (Rodríguez Lombana, 2019).

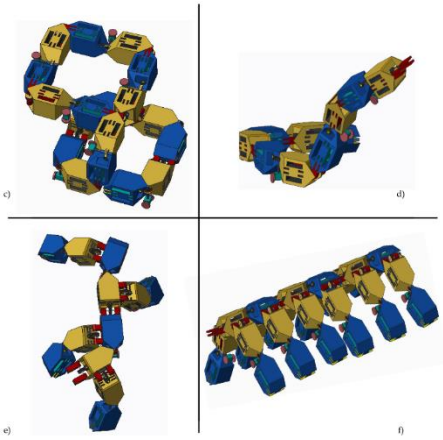


Figura 11. Robot tipo hibrido HexaMob. Tomado de (Reddy et al., 2017).

5.2.1.4. Robot modular EMERGE

El robot modular EMERGE es una plataforma de hardware abierta diseñada para realizar experimentos de aprendizaje en el área de la locomoción y la evolución. Con su capacidad de ensamble rápido, este robot puede adoptar diversas morfologías que pueden estar equipadas con sensores básicos. Cada módulo que compone la estructura del robot está diseñado como un cubo con una bisagra central, como se muestra en la Figura 12. En esta bisagra se encuentra un servomotor Dynamixel AX-A12A, con soportes atornillados al eje del motor y a su parte inferior. Cada módulo está compuesto por cuatro caras impresas en 3D, cada una de las cuales alberga dos pares de imanes de neodimio, permitiendo la conexión entre módulos por medio de estos. Además, cada módulo se compone de tres caras hembras y una macho, lo que permite tanto la conexión física como la transmisión de la alimentación a través de la unión de una cara macho con una hembra (Moreno et al., 2017).

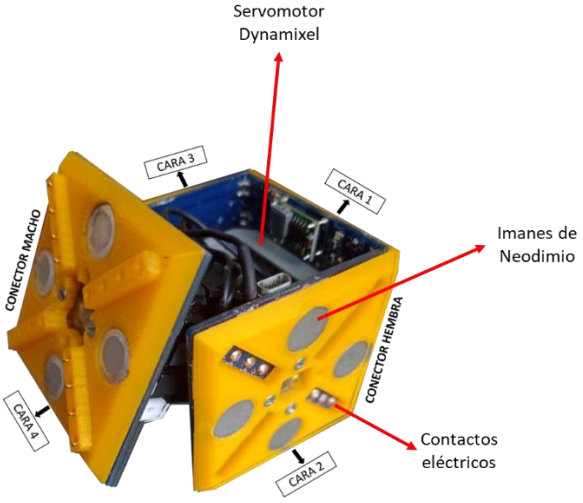


Figura 12. Modulo del robot EMERGE. Elaboración propia.

La conexión entre una cara macho y una hembra puede soportar hasta 1,3 Nm de torque antes de desconectarse por completo. Cada motor de los módulos tiene un torque máximo de 2,5 Nm y unas dimensiones de 32 x 50 x 40 mm. Además, cuenta con una velocidad de transmisión de datos que oscila entre 7,843 bps y 1 Mbps con un margen de error inferior al 3%, como se muestra en la Tabla 1. El rango de alimentación de entrada del motor va desde 9V hasta 12V. Para el caso de múltiples motores conectados, es necesario establecer un valor de identificación del motor en un rango de 0 a 253 bits para enviar las instrucciones correctamente al robot. Es posible enviar paquetes simultáneamente a todos los motores conectados identificando uno de ellos como el 254, lo que garantiza la sincronización de las instrucciones. Además, estos motores tienen la capacidad de sensar la posición, temperatura, carga, voltaje de entrada y otros parámetros a los que están expuestos (ROBOTICS, n.d.).

Tabla 1. Determinación de la velocidad de comunicación serial entre un controlador y el DYNAMIXEL a partir de la velocidad de transmisión. Tomado de (ROBOTICS, n.d.).

Value	Baud Rate (bps)	Margin of Error
1 (Default)	1M	0.000%
3	500,000	0.000%
4	400,000	0.000%
7	250,000	0.000%
9	200,000	0.000%
16	115200	-2.124%
34	57600	0.794%
103	19200	-0.160%
207	9600	-0.160%

Otro de los parámetros dinámicos de estos servomotores es la posición objetivo, la posición objetivo del servomotor varía de 0 a 1023 bits y cada unidad equivale a 0.29° de rotación, este motor permite la rotación en dirección a las manecillas del reloj y en dirección contraria a estas, por ejemplo, si la posición objetivo es 1023 este rotará en dirección opuesta a las manecillas del robot como se ve en la Figura 13. Dada la posición objetivo, existe un parámetro que es la velocidad de movimiento a la posición objetivo, esta varía dependiendo el modo de operación del robot, ya sea modo conjunto o rueda; para modo conjunto el parámetro es de 0 a 1023 bits, por cada unidad la velocidad es alrededor de 0.111rpm, si el valor es fijado en 0 bits significa que el motor está trabajando sin control de velocidad, si el valor es fijado en 1023 bits la velocidad para que llegue a la posición objetivo es de 114rpm (ROBOTICS, n.d.).

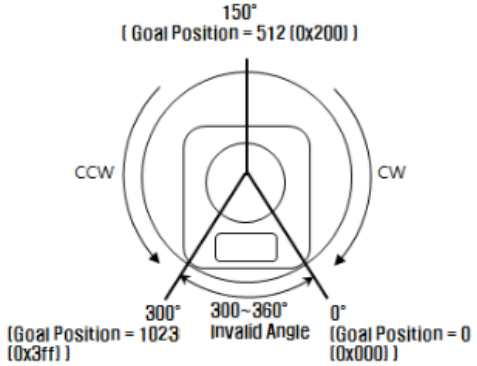


Figura 13. Vista frontal DYNAMIXEL, detalle posición objetivo. Tomado de (ROBOTICS, n.d.)

5.2.2. Control de movimiento de robots modulares ápodos

En la robótica, el sistema de control es el encargado de ejecutar y procesar las acciones, la locomoción y los procesos lógicos del robot. Existen dos tipos de sistemas de control para los robots modulares, el control centralizado que se basa en un controlador principal ensamblado en uno de los módulos del robot, este controlador se comunica con todos los actuadores y sensores de cada módulo del robot (Jiménez Cámara, 2012). El otro sistema de control es el descentralizado el cual funciona con el ensamble de dos o más controladores acoplados a sus correspondientes módulos, en esta red la comunicación es entre todos los controladores que representan al módulo en el cual están conectados (Nascimento Martins et al., 2008).

5.2.2.1. Generadores sinusoidales

Esta técnica de control de movimiento se basa en realizar una ecuación sinusoidal, entregando el valor de la posición de un módulo, a través de la estimación de la posición de los demás módulos se genera un desfase respectivo a cada uno de éstos dando como resultado el movimiento sinusoidal. Este tipo de control de movimiento ha sido ampliamente aplicado en morfologías tipo cadena puesto que permite que el movimiento sinusoidal se genere armónicamente, por ser una estructura unidimensional, facilitando el recorrido del robot, es una técnica de control simple y adaptable a superficies rugosas (Ellips Masehian, 2015)(Rodríguez Lombana, 2019)(González Gomez, 2008).

En la ecuación (1) se puede ver la representación del modelo general que implementa el controlador, donde x es el valor de la posición del módulo, A el valor de la apertura máxima que el actuador puede tener en cada módulo, θ el ángulo que varía en el tiempo con respecto al recorrido de cada módulo y φ el desfase que se le asigna a cada módulo dependiendo de la posición en la que se encuentre en la morfología (Ellips Masehian, 2015), (González Gomez, 2008), (Alberto Brunete et al., 2017).

$$x = A * \sin(\theta * t + \varphi) \tag{1}$$

5.2.2.2. Cinemática inversa

Esta técnica de control de movimiento se basa en el uso de ecuaciones cinemáticas con el objetivo de generar el movimiento en cada actuador generando la locomoción del robot, cuantos más actuadores o módulos tenga el robot el modelo cinemático de movimiento se tornará más complejo. El lazo de control implementado en este método es de lazo abierto, puesto que el setpoint de la trayectoria se va interpolando con la finalidad de que el robot realice un recorrido de pasos continuos (Felipe et al., 2006), (Vele G, 2005), (Ibarra Peñaranda et al., 2021).

5.2.2.3. Tablas de control de movimiento

Esta técnica consiste en que todos los parámetros de la rutina de movimiento se encuentran registrados jerárquicamente en un vector que alberga las respectivas posiciones de cada articulación. Seguidamente, el controlador toma ordenadamente los datos almacenados en el vector y envía las órdenes a los actuadores, una vez el actuador se sitúa en el punto determinado, el controlador enviará el siguiente comando del vector, cuando finaliza todos los parámetros del vector, vuelve al punto inicial del vector y ejecuta nuevamente la misma rutina (Ellips Masehian, 2015), (Moreno García, 2010).

5.2.2.4. Generador central de patrones (CPG)

Es una técnica inspirada en los movimientos rítmicos presentes en los animales, el CPG en muchas implementaciones está basado en redes neuronales, por medio de la réplica de movimientos produce patrones de movimiento coordinado o locomoción con el objetivo de generar un desplazamiento (Alberto Brunete et al., 2017). Esta técnica permite que configurando los parámetros del CPG se obtenga una variedad de movimientos y patrones rítmicos. La estructura de este sistema de control de movimientos está basado en un grupo de redes que están interconectadas entre sí, con el objetivo de generar una rutina de tareas repetitivas (Ellips Masehian, 2015), (Zamorano Martín, 2011), (Moreno García, 2010), (González Gomez, 2008).

5.2.3. Inteligencia Artificial

La inteligencia artificial o como se le conoce en ingles *artificial intelligence,* es una rama de los algoritmos computacionales la cual se creó para simular o asemejar la inteligencia humana mediante el aprendizaje del entorno que lo rodea por medio de la experiencia o los datos de ejemplo como lo pueden ser las bases de datos. El principal énfasis de esta rama es la creación de algoritmos que puedan autónomamente incluir modelos sin la intervención o asistencia humana. Estos algoritmos de aprendizaje funcionan mediante la generación automática de programas o modelos a partir de datos, de esta manera resuelven distintos problemas, estos modelos pueden ser predictivos para descifrar el futuro, como por ejemplo el comportamiento de un virus o ser descriptivo para ganar conocimiento a partir de los datos (Reyes García et al., 2018), (Alpaydin, 2014).

Estos algoritmos son clasificados en tres grandes categorías que son aprendizaje supervisado, aprendizaje no supervisado y aprendizaje de refuerzo, como se ve en la Figura 14, de cada una de estas tres grandes ramas se desprenden los diferentes algoritmos como modelos geométricos, modelos probabilísticos y modelos lógicos; así como, algoritmos que crean descripciones o resúmenes, algoritmos que segmentan/agrupan datos automáticamente, sistemas que crean automáticamente modelos para predecir salidas, sistemas que encuentran automáticamente dependencias entre variables, y sistemas que aprenden a crear planes o estrategias de control para resolver problemas de decisión secuenciales. A continuación se abordaran las distintas clasificaciones de estos algoritmos mencionadas anteriormente (Reyes García et al., 2018), (Alpaydin, 2014).

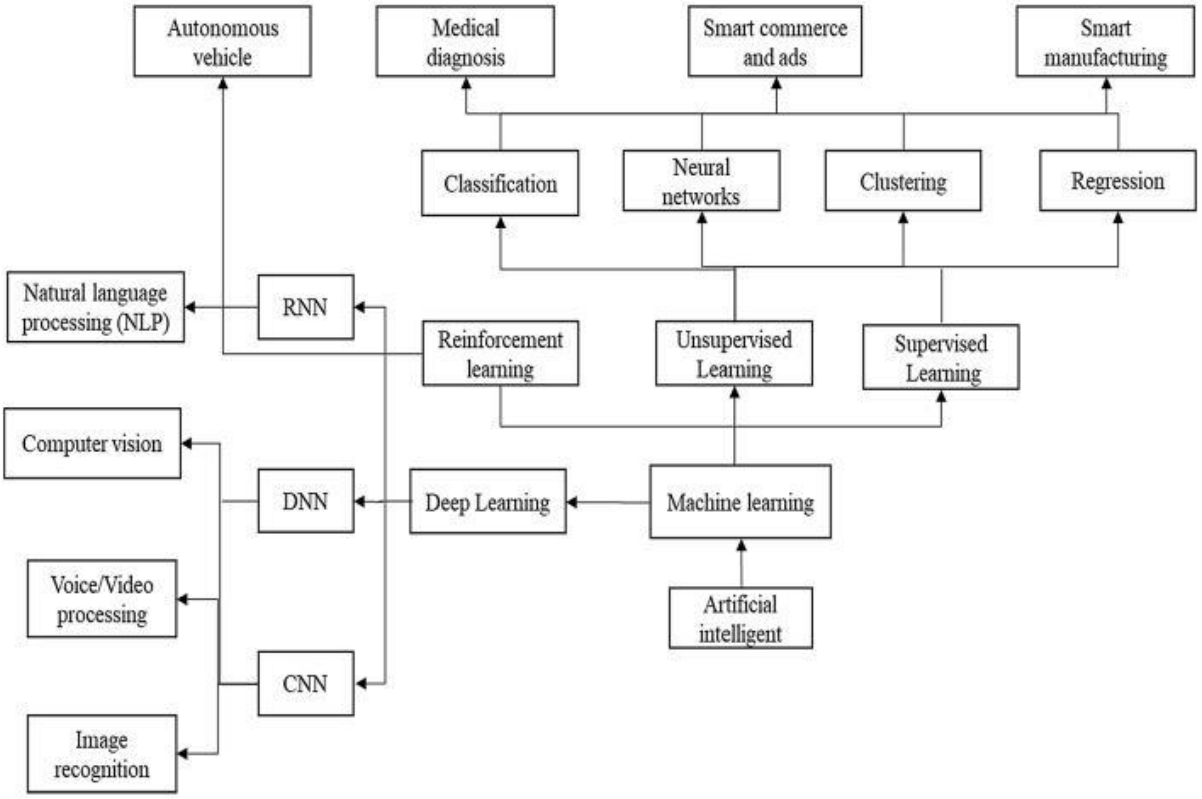


Figura 14. Clasificación de las metodologías de inteligencia artificial (Trappey et al., 2020).

5.2.3.1. Machine learning

El aprendizaje automático o machine learning en inglés es una rama de la inteligencia artificial que se enfoca en el uso de datos y algoritmos para imitar el aprendizaje humano, gradualmente para mejorar su asertividad. Esta rama, es muy importante en el campo de la ciencia de datos, por medio del uso de métodos estadísticos los algoritmos son entrenados para clasificar o predecir, algunos de los algoritmos más reconocidos son las redes neuronales, regresión lineal, regresión logística, clustering o agrupamiento, arboles de decisión y bosques aleatorios. Como se puede observar en la Figura 14 el aprendizaje profundo es una rama del aprendizaje autónomo (H. H. Martínez et al., 2020).

5.2.3.2. Aprendizaje profundo

El deep learning o aprendizaje profundo se caracteriza por estar conformado por capas de algoritmos que permiten traducir o comprender mejor los datos. En este caso los algoritmos no hacen regresiones básicas, sino que tiene capas de algoritmos no lineales que crean representaciones distribuidas, algunos de las implementaciones de este tipo de aprendizaje son en la clasificación de videos e imágenes, reconocimiento de voz, entre otros (Amazon Web Services (AWS), n.d.).

Para la implementación de este tipo de aprendizaje, lo más común es utilizar el lenguaje de programación Python, el cual es un lenguaje de código abierto, es decir, que sus librerías y ambientes de programación los puede utilizar cualquier persona. Además de este lenguaje de

programación, también es necesario implementar una librería que permita realizar un aprendizaje profundo como lo son PyTorch, TensorFlow y Keras, esta última funciona como una interfaz que internamente usa otras librerías como TensorFlow, CNTK o Theano (Martinez Heras, 2020).

La implementación de una API como keras permite aumentar el número de neuronas y capas a una red neuronal fácilmente, así como la configuración de parámetros para el entrenamiento de la inteligencia artificial, como lo son los optimizadores, encargados de mejorar el modelamiento de los parámetros de la red neuronal para reducir el error; otro de los parámetros que se deben tener en cuenta es la pérdida, que hace referencia a la función que mide la precisión del modelo durante el entrenamiento para dirigir el modelo hacia la dirección adecuada (Martinez Heras, 2020).

5.2.3.3. Modelos de aprendizaje supervisado

Este modelo funciona bajo la modalidad de entradas tipo datos patrones o eventos pasados junto con las correspondientes respuestas a esas entradas, la función del modelo es predecir la respuesta una vez ha estudiado el patrón de relación de las entradas con las salidas, este pronóstico se basa en un ajuste de pesos para que la predicción sea buena y el error vaya disminuyendo, de esta manera se evita realizar análisis y modelos matemáticos de alta complejidad donde muchas veces los datos no pueden tener una ecuación que les relacione (Matich, 2001), (Bhattacharya et al., 2018), (Behal et al., 2009), (Martín del Brío & Sanz Molina, 2001). Dentro de este tipo de modelos se encuentran los algoritmos de regresión como el SVM, agrupamiento, redes neuronales y de clasificación.

5.2.3.4. Modelos de aprendizaje no supervisado

El modelo está basado en enseñarle al algoritmo los posibles datos que va recibir, pero no se le indica cuales son las salidas así que el algoritmo debe mapear los datos hallando la similitud de los datos de entrada y los patrones que son más frecuentes para esas entradas, este tipo de comportamientos es llamado en la estadística como estimación de densidad (Maglogiannis et al., 2021), (Alpaydin, 2014), (Martín del Brío & Sanz Molina, 2001). Dentro de este tipo de modelos se encuentran los algoritmos de regresión como el SVM, agrupamiento, redes neuronales y de clasificación, estos algoritmos pueden ser tanto con los modelos supervisados como no supervisados.

5.2.3.5. Modelos de aprendizaje por refuerzo

La idea principal de este modelo es desarrollar una técnica en la cual se llegue a la mejor respuesta o destino sin conocer las acciones intermedias, está basado en la idea de pequeñas acciones llevan a mejores o diferentes resultados como en los juegos de táctica (Reyes García et al., 2018), (Alpaydin, 2014), (Martín del Brío & Sanz Molina, 2001). Este tipo de aprendizaje se enfoca en encontrar la combinación de acciones óptimas para la resolución del problema, aprendiendo del historial de acciones y optimizando el aprendizaje. Este modelo es implementado en los automóviles autónomos, así como en el control de semáforos viales inteligentes (Mannion et al., 2016).

5.2.3.6. TensorFlow-Keras

TensorFlow es una plataforma – librería de código abierto que permite la creación y uso de modelos de machine learning desarrollada por Google, esta permite múltiples niveles de abstracción de datos dependiendo la aplicación y necesidades del usuario; la construcción y entrenamiento de modelos

se hace por medio de la interfaz de programación de aplicación API de keras, lo que facilita la introducción en el machine learning; Keras es una API que sigue las mejores prácticas para la reducción de la carga cognitiva y está diseñada para humanos, lo que permite minimizar la cantidad de acciones que deben ejecutar los usuarios para elaborar y entrenar modelos en los casos más comunes. Algunas de los módulos que tiene esta API son módulos de funciones de activación, módulos de aplicaciones que tienen pesos predeterminados, un módulo de Backend, módulo de pequeños conjuntos de datos para experimentar, módulos de estimadores, módulos de capas, perdidas, métricas, módulo de modelos que permite la creación, configuración, clonación, carga y guardado de modelos; módulos de optimizadores prefabricados, entre otros (TensorFlow - Google, n.d.), (Keras, n.d.).

5.3. Marco Legal

Es esta sección se describen las normas que intervienen en el desarrollo de este proyecto.

5.3.1. ISO/IEC AWI TS 8200

Es una norma en desarrollo que define el marco de referencia básico con principios, características y enfoques para la realización y mejora del control de los sistemas automatizados por inteligencia artificial (ISO/IEC, n.d.).

5.3.2. ISO/IEC TR 24029-1:2021

Es una primera parte de la norma que define la evaluación de la robustez de las redes neuronales, en esta se hace un marco de referencia y expone los antecedentes de los métodos existentes para la evaluación de robustez de las redes neuronales (ISO - ISO/IEC TR 24029-1:2021 - Artificial Intelligence (AI) — Assessment of the Robustness of Neural Networks — Part 1: Overview, 2021).

6. Metodología

En esta sección, se describe el procedimiento que se realiza para la elaboración del algoritmo de aprendizaje computacional para el control de movimiento del robot modular EMERGE, el cual comprende el desarrollo del algoritmo con el tipo de control de movimiento de tablas de movimiento, posteriormente se expone el algoritmo con el tipo control de movimiento de aprendizaje computacional propuesto.

6.1. Control de movimiento del robot EMERGE con tablas de movimiento

Inicialmente, se implementa el control de movimiento basado en tablas de movimiento como se ve en el diagrama de la Figura 15, donde se observa la implementación de una interfaz gráfica que se ve en la Figura 16, basada en el software elaborado por (Morantes et al., 2020), con la finalidad de elaborar los movimientos de avanzar y girar, en las morfologías cadena, tipo ELE y bloque. Posteriormente se ve la comunicación serial de la Raspberry con el Arduino para la transmisión de los comandos elaborados en la interfaz gráfica en la Raspberry Pi y posterior envío a cada uno de los motores de los módulos por medio de comunicación serial.

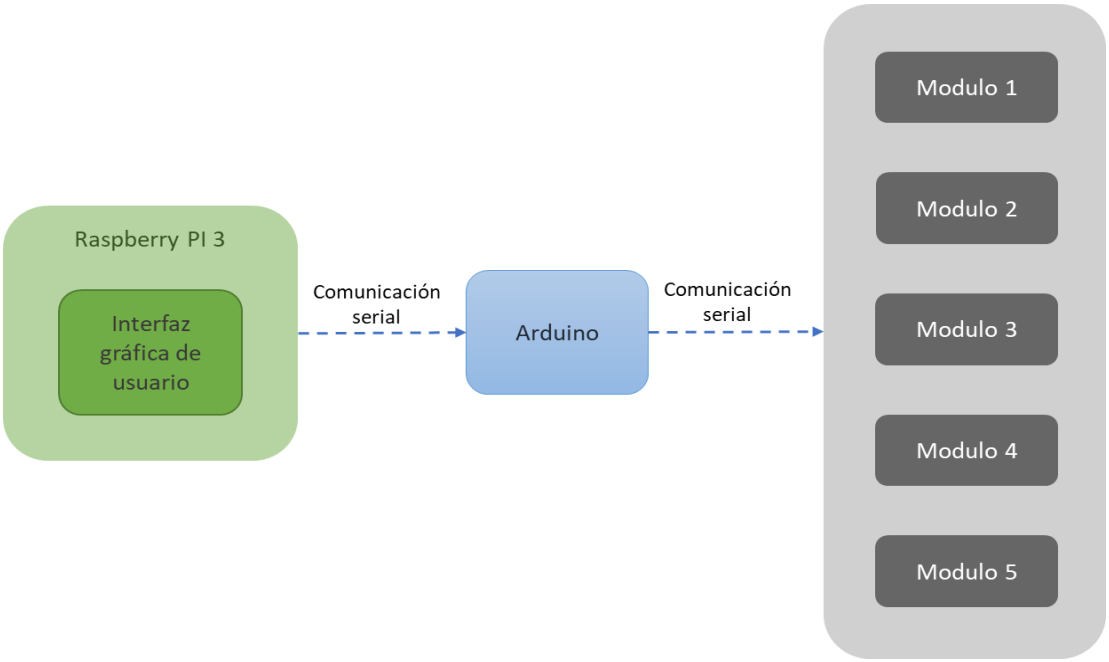


Figura 15. Diagrama funcionamiento elaboración tablas de movimiento. Elaboración propia.

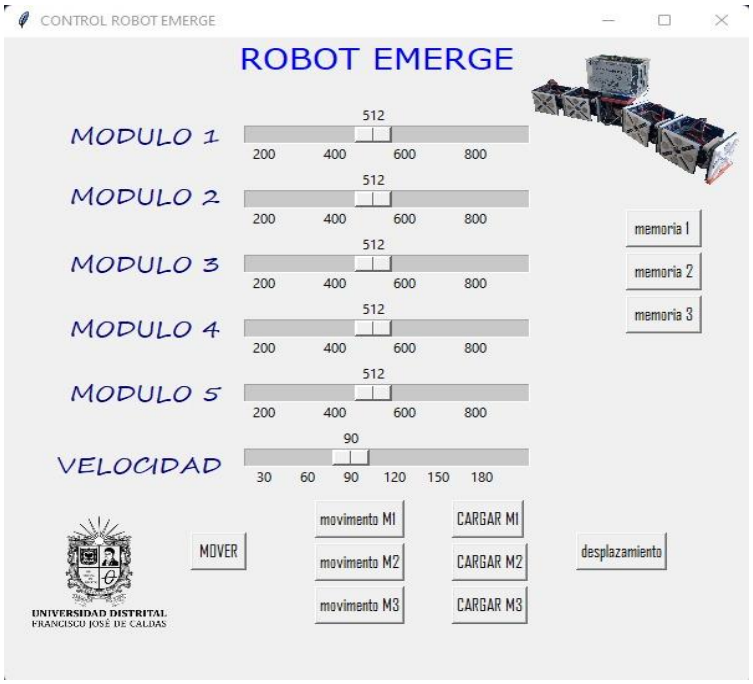


Figura 16. Interfaz gráfica de entrenamiento de tablas de movimiento. Elaboración propia.

En la Figura 17 se puede observar la configuración física de los módulos en las tres morfologías para realizar el entrenamiento de las tablas de movimiento.

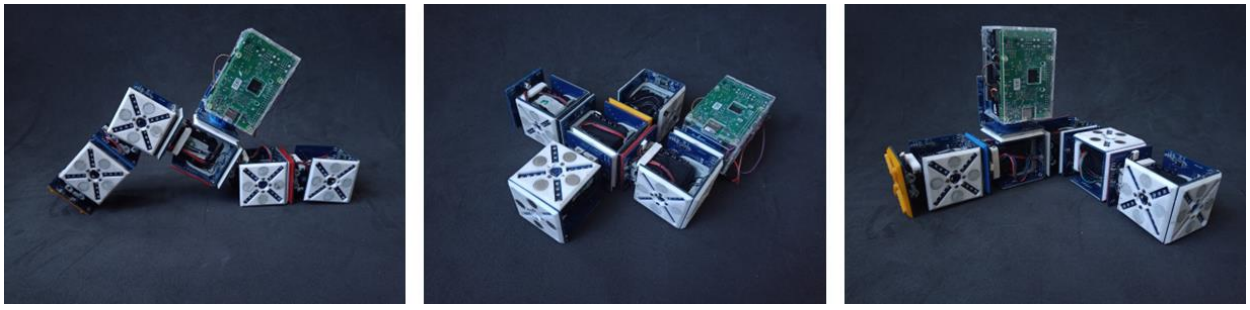


Figura 17. Robot EMERGE morfologías cadena, bloque y tipo ELE respectivamente, para el entrenamiento de tablas de movimiento. Elaboración propia.

En la Raspberry Pi 3 se implementa un algoritmo para el entrenamiento de movimientos de los módulos que componen el robot EMERGE, este algoritmo está compuesto por una interfaz gráfica, por medio de esta se envían los vectores de movimiento al Arduino a través de comunicación serial y con el mismo tipo de comunicación se envía la información a cada uno de los módulos, la estructura de los vectores enviados al Arduino está compuesta por un carácter de inicio de trama, el parámetro de velocidad, parámetros de movimiento de los 5 motores y un carácter de fin de trama, como se ve en la Figura 18; una vez se envía haciendo clic en el botón de mover y se obtiene el movimiento deseado este se guarda en un archivo .txt haciendo clic en alguno de los botones de memoria, cuando se obtiene una rutina adecuada para avanzar o girar se ejecutan todos los vectores guardados anteriormente por medio del botón de movimiento M#. Por último, se prueba si la rutina

de las memorias se ejecutó bien por medio del botón de desplazamiento. Esto se repite para las tres morfologías, cadena, tipo ELE y bloque.

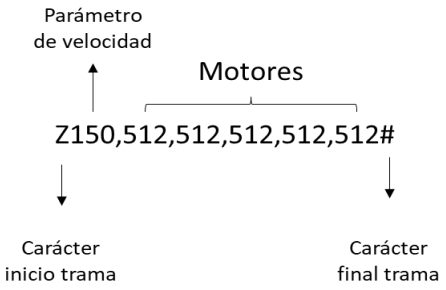


Figura 18. Parámetros de trama al controlador de motores Arduino. Elaboración propia.

Seguidamente, se elabora un controlador local encargado de la elaboración de la transmisión de los movimientos basado en tablas hacia los módulos como se ve en la Figura 19; el controlador local está compuesto por una Raspberry Pi 3, un Arduino, dos sensores ultrasónicos y dos reguladores de voltaje para la alimentación tanto de la Raspberry como del Arduino Uno junto con los sensores como se ve en la Figura 20, estos componentes se comunican con protocolo serial como se ve en la Figura 19, en esta figura también se observa que los módulos están en comunicación con el Arduino por medio de protocolo serial. La Raspberry contiene las tablas de movimiento elaboradas anteriormente, aquí también se ejecuta el algoritmo de control que es retroalimentado por las señales de los sensores ultrasónicos conectados al Arduino el cual se encarga de procesar las señales de los sensores y enviarlas a la Raspberry.

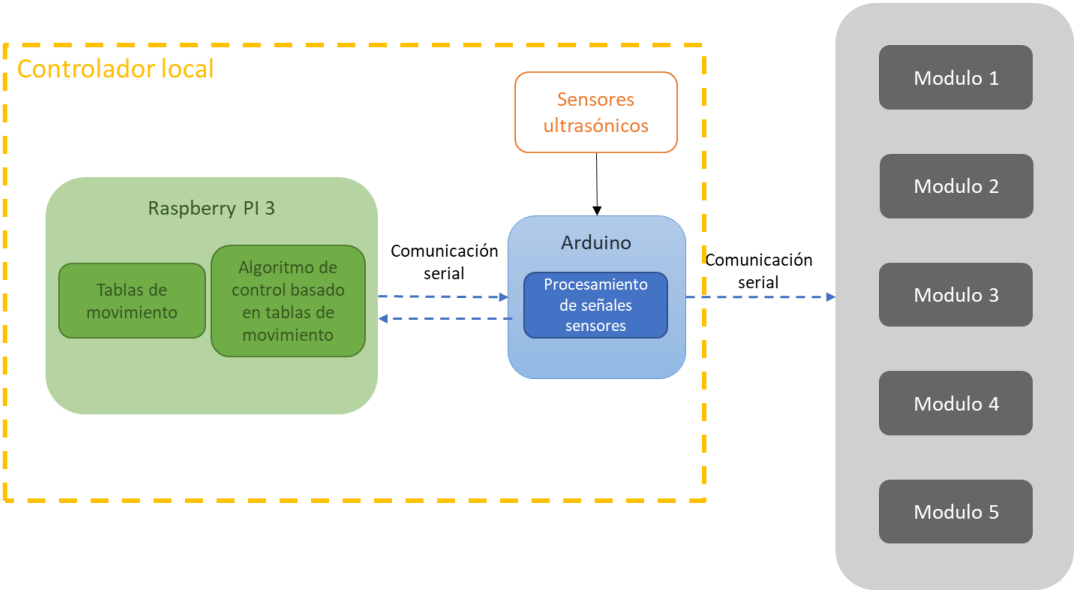


Figura 19. Diagrama de control por tablas de movimiento. Elaboración propia.

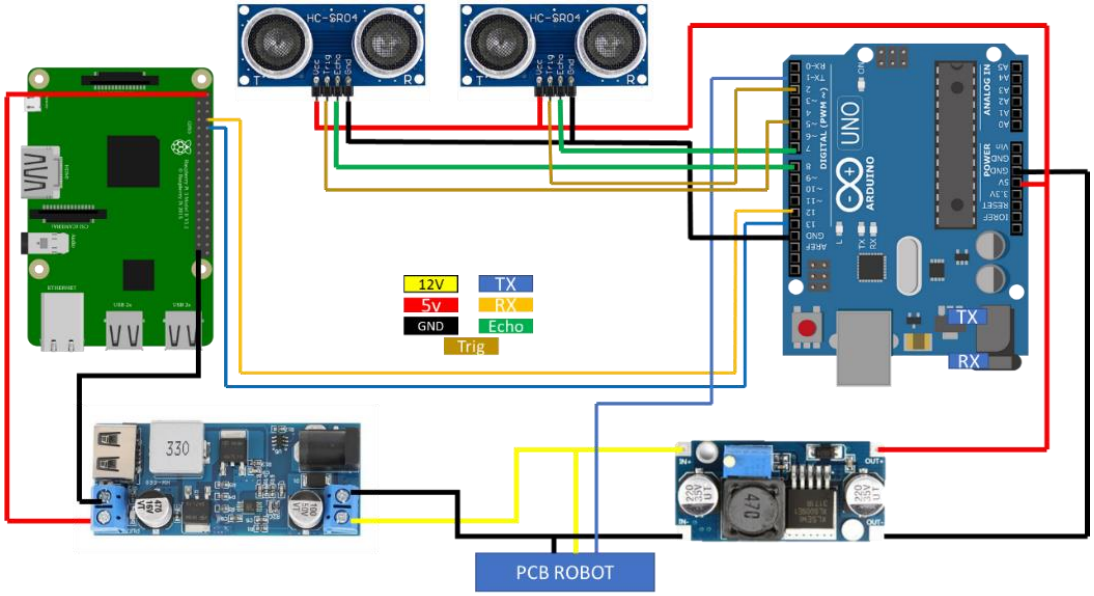


Figura 20. Diagrama electrónico de conexión del controlador local. Elaboración propia.

El algoritmo de control basado en tablas de movimiento se puede observar en la Figura 21, en el cual se hace una petición al Arduino para obtener la señal de los sensores ultrasónicos encargados de medir la distancia a la que puede estar un obstáculo si lo detecta, posteriormente realiza una binarización por cada sensor, en este caso 2 sensores, si la señal detectada es menor o igual a 40cm se marca como que el sensor detectó un obstáculo en el lateral del sensor que se está binarizando y se guardan estos valores para posteriormente determinar si el motor debe avanzar, girar a la izquierda o a la derecha según donde se encuentre el obstáculo. Este algoritmo de control tiene en cuenta la morfología que se le especifique, aplica la misma lógica de control para cadena, tipo ELE y bloque.

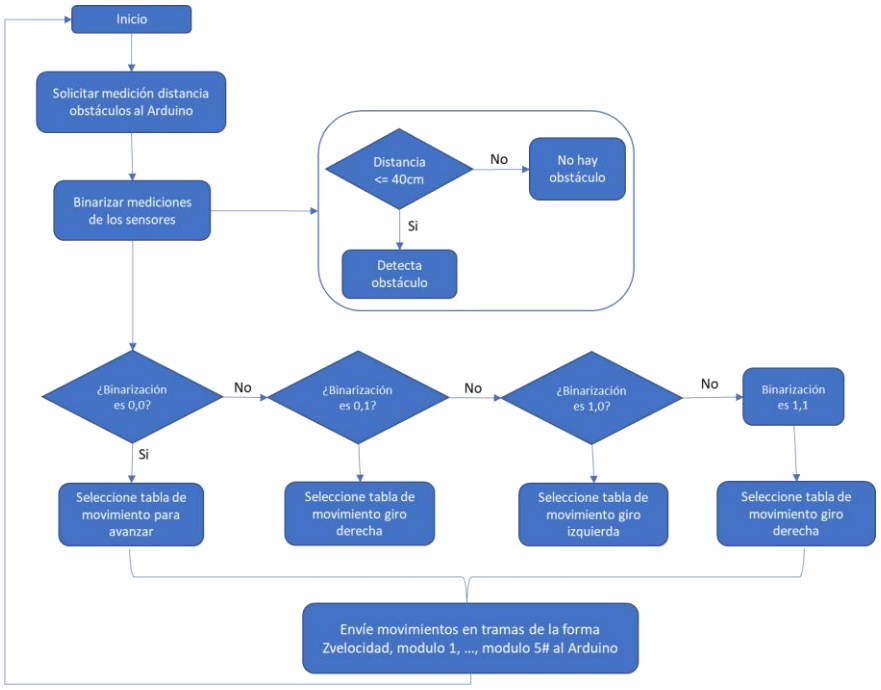


Figura 21. Pseudocódigo algoritmo de control de movimiento por tablas de movimiento. Elaboración propia.

Por otra parte, en el controlador local también está el Arduino Uno, que está conectado con los sensores ultrasónicos, este contiene un algoritmo el cual se observa en la Figura 22, este captura los datos de las mediciones del sensor cuando la Raspberry se lo solicita y convierte este valor de bits a centímetros para el envío de estos datos, también en este algoritmo se encuentra la reconstrucción de la trama proveniente de la Raspberry para el movimiento de los módulos; igualmente se encarga de la comunicación con los servomotores de cada módulo. La activación de cada función explicada anteriormente está dada por medio de banderas dependiendo del estado del proceso, uno de estos es la solicitud de lectura de sensores y el otro es la transmisión de comandos a los módulos.

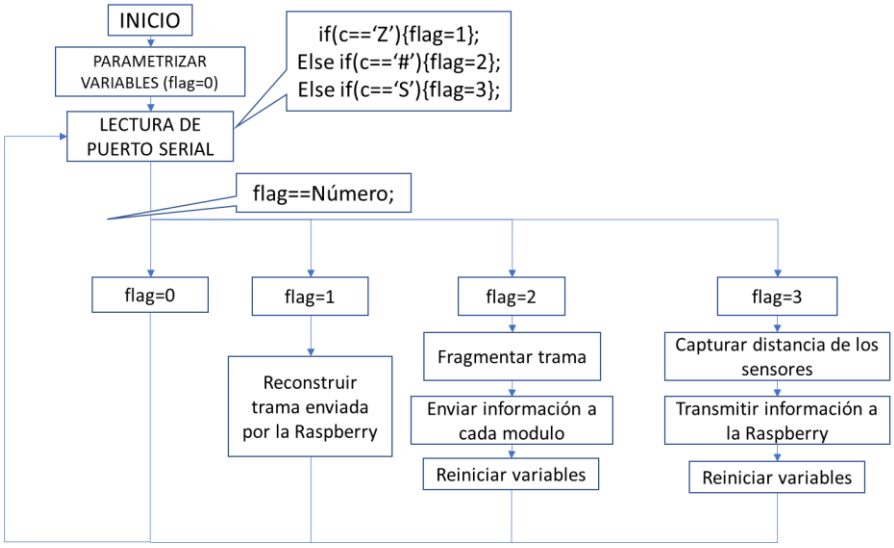


Figura 22. Pseudocódigo del algoritmo implementado en el Arduino. Elaboración propia.

6.2. Desarrollo algoritmo de control de movimiento con aprendizaje computacional

Para la elaboración del algoritmo de control se toma el mejor patrón de movimiento, tanto para avanzar como para girar según los estados de los sensores especificados anteriormente y que también se pueden ver en la Figura 23 (color naranja); dentro del modelo de entrenamiento se encuentra el algoritmo de aprendizaje basado en redes neuronales que está diseñado como se presenta en la Figura 23 (color azul) y en la Figura 24. Por último, se encuentra el resultado que son los modelos para cada tipo de estado de los sensores para cada una de las morfologías.

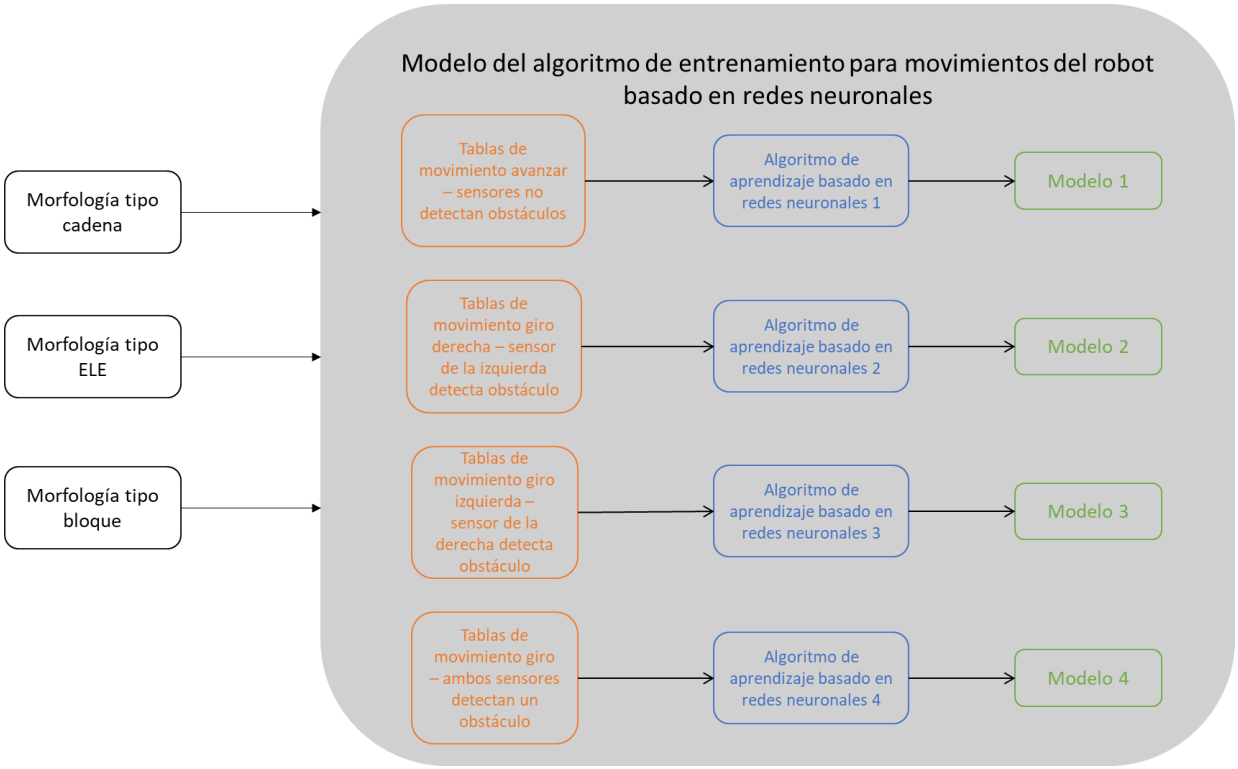


Figura 23. Diagrama de la solución implementada para entrenamiento redes neuronales para los modelos de movimiento. Elaboración propia.

El algoritmo de aprendizaje basado en redes neuronales descrito en la Figura 24, empieza con la lectura de las mejores tablas de movimiento obtenidas en el experimento de la sección 6.1, se convierten en datos legibles para la red neuronal, se establecen los datos de entrada de la red que son los estados de los sensores, después se configuran los parámetros de la red, aquí se tienen en cuenta el número de entradas y salidas a partir de la cantidad de movimientos en cada tabla y dependiendo cada morfología, este parámetro no es estandarizable dado que por ejemplo, el movimiento para avanzar en morfología cadena tiene menor cantidad de movimientos que la morfología tipo ELE; una vez configurados los parámetros de modo de activación, cantidad de capas de redes ocultas, perdida, optimización, épocas y tamaño del lote, se procede a realizar el entrenamiento de la red, guardar el modelo obtenido y validar el mismo hasta encontrar la configuración de estos parámetros que permita obtener el modelo correspondiente a la tabla con la que se le enseñó.

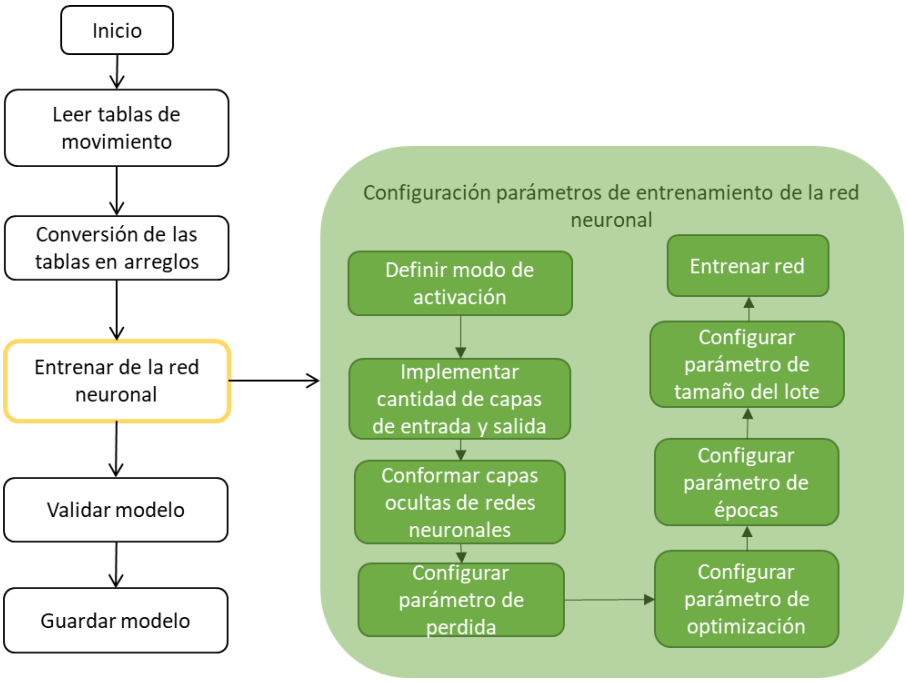


Figura 24. Algoritmo de entrenamiento de movimientos basado en redes neuronales. Elaboración propia.

Para desarrollar el algoritmo de control de movimiento con aprendizaje computacional, se llevan a cabo diferentes enfoques para entrenar las redes neuronales multicapa y se selecciona el que permite obtener los modelos de redes neuronales óptimos para el control de movimiento del robot.

El primer enfoque, consiste en implementar la librería Scikit-Learn en Python, para el entrenamiento de la red neuronal a partir de las mejores tablas de movimiento obtenidas de la sección 6.1. Sin embargo, en este enfoque el tipo de datos de entrenamiento del modelo no es compatible con los datos de entrenamiento esperados por la librería, puesto que para este entrenamiento se está implementando una lista con múltiples valores de movimiento que representan las tablas de movimiento.

En el segundo enfoque, se realiza el entrenamiento mediante la implementación de la librería Tensorflow-Keras. Este programa se encuentra en el Anexo 3, donde se puede observar que es necesario implementar cientos de capas ocultas con diferentes modos de activación para uno de los cuatro posibles movimientos de las morfologías cadena, bloque o tipo ELE. Bajo este enfoque, se logra entrenar las redes neuronales de las tres morfologías. Sin embargo, la complejidad del algoritmo que se ejecuta en un computador local, con tiempos de entrenamiento de 20 minutos, no permite una experimentación óptima, por lo cual se opta por implementar un nuevo algoritmo.

Para el tercer enfoque, se implementan los algoritmos que se encuentran en el Anexo 4 y Anexo 5. Este se ejecuta en Google Colab, una herramienta gratuita de Google que permite la ejecución de programas de Python en la nube, junto con una GPU de Google Compute Engine, 12 GB de memoria RAM y 68 GB de memoria ROM. Además, ofrece la posibilidad de entrenar y guardar modelos entrenados en diferentes versiones de tensorflow. En este enfoque, se implementa la

versión 2.5 de tensorflow, lo que permite que el entrenamiento sea más rápido y que los ajustes se puedan realizar en menos de 20 minutos en comparación con el método anterior. Además, el algoritmo requiere una cantidad significativamente menor de entrenamiento para las 13 redes neuronales, logrando finalmente los primeros 13 modelos de redes entrenadas.

Finalmente, en el cuarto enforque, se implementan los mismos algoritmos del Anexo 4 y Anexo 5, pero se realiza el entrenamiento en la Raspberry pi3 del controlador central del robot. Esto se debe a que actualmente Google Colab ya no soporta la versión 2.5 de tensorflow, y el algoritmo de control solo puede leer modelos que estén con esta versión. Para la implementación de tensorflow en la Raspberry pi3, es necesario crear un entorno virtual de Python para la instalación de la librería y la ejecución de los algoritmos de entrenamiento.

Una vez se prueba el cuarto enfoque con el entrenamiento de los modelos de movimiento del robot, se crea el algoritmo de entrenamiento del modelo de los sensores, el modelo que se obtiene en este algoritmo determina el modelo de movimiento que se debe enviar al Arduino para que los módulos ejecuten los respectivos patrones. Este algoritmo se entrena a partir de una lista de valores que representan cada uno de los posibles estados del par de sensores asociados a la respectiva morfología como se ve en la Figura 25, para la morfología cadena los posibles estados están representados por los valores que inician en 1, para la morfología tipo ELE son los que inician en 2 y para la morfología bloque son los que empiezan por el 3, como como se puede ver en el Anexo 5; los valores obtenidos de este entrenamiento son etiquetas que corresponden a los modelos obtenidos del entrenamiento de los movimientos.

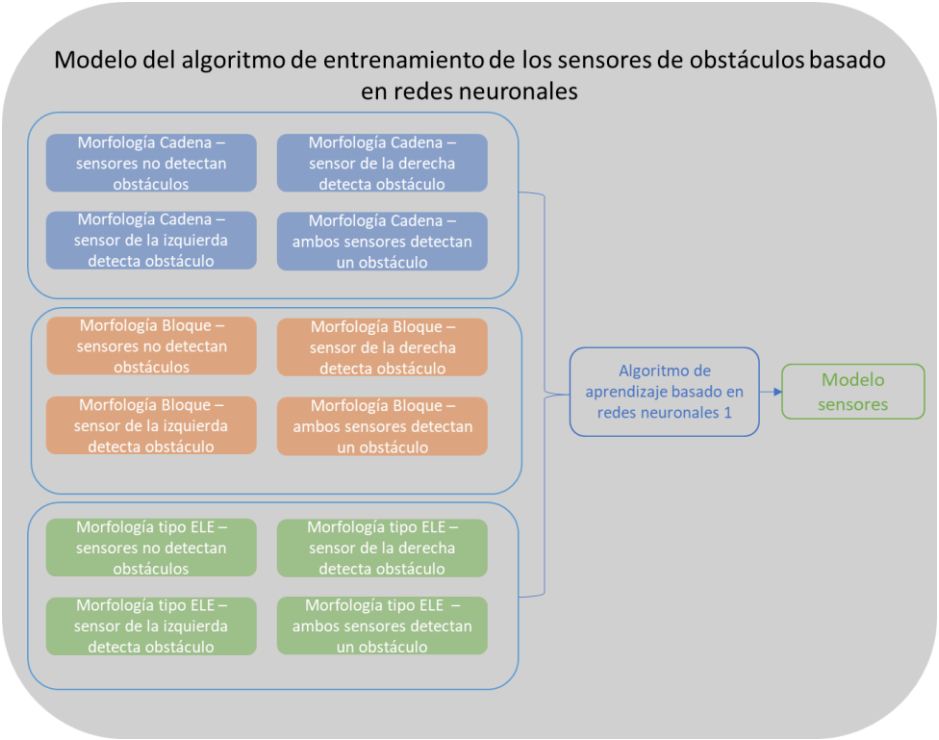


Figura 25. Diagrama de la solución implementada para el entrenamiento de la red neuronal para el modelo de los sensores. Elaboración propia.

6.3. Implementación algoritmo de control basado en aprendizaje computacional

Para finalizar, en la Figura 26 se puede ver la implementación del algoritmo de control basado en aprendizaje computacional que se encuentra en color verde, en este se encuentran cargados los modelos obtenidos de la sección anterior, los cuales son los modelos de movimiento y el modelo de los sensores. Para la ejecución de este algoritmo es necesario la implementación de la librería TensorFlow – Keras para la carga de los modelos anteriormente mencionados. Cabe recordar que este algoritmo se ejecuta en la Raspberry bajo el mismo modelo de la Figura 19 y Figura 20.

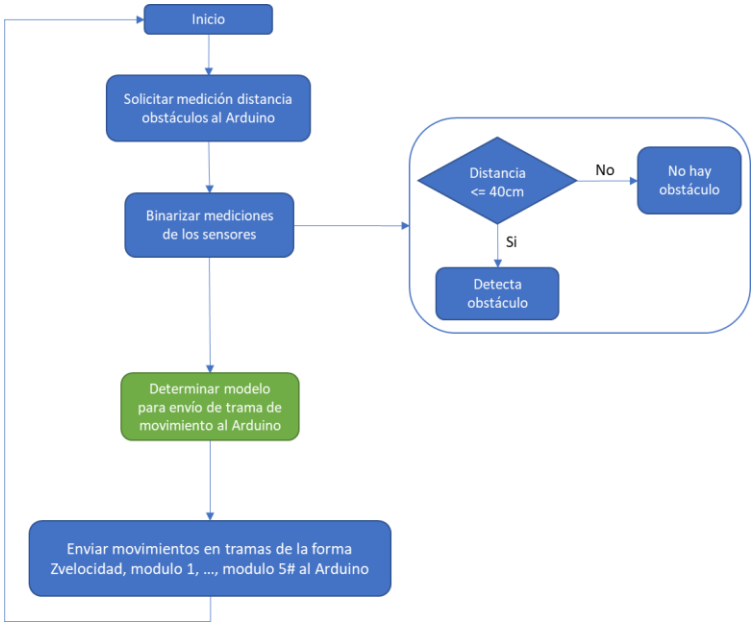


Figura 26. Pseudocódigo del algoritmo de control de movimiento por aprendizaje computacional. Elaboración propia.

Para la implementación de este algoritmo se crea una interfaz gráfica como se ve en la Figura 27, la cual permite cargar los modelos entrenados mencionados en la sección 6.2 haciendo clic en el botón “cargar redes”, probar que los módulos estén funcionando correctamente como se ve en la ventana emergente que se activa al hacer clic en el botón “test”. Para seleccionar la morfología configurada en el robot para que ejecute el algoritmo de la Figura 26 se hace clic en el nombre de la morfología correspondiente; en la Figura 28 se observa el cambio del gráfico según se seleccione la morfología y de esta manera internamente en el programa se definen los movimientos que se deben ejecutar como se ve en el Anexo 6.

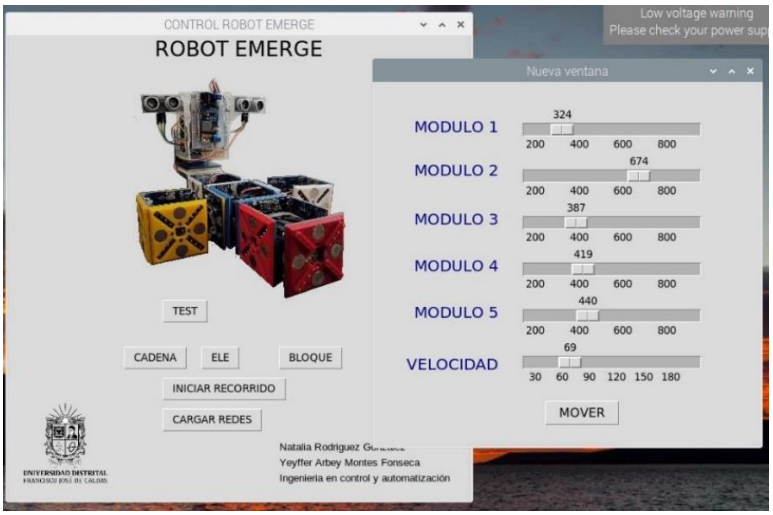


Figura 27. Interfaz gráfica activación redes neuronales. Elaboración propia.

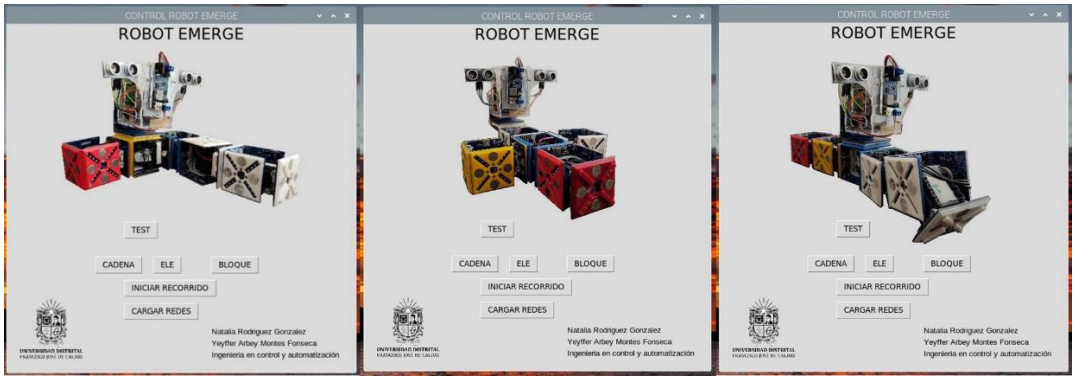


Figura 28. Representación de la interfaz gráfica por tipo especifico de morfología. Elaboración propia.

7. Resultados

En esta sección, se presentan los resultados obtenidos en la elaboración de la estructura física del robot EMERGE en las morfologías cadena, tipo ELE y bloque. También se presentan los resultados de las pruebas con la técnica de control de movimiento basado en tablas de movimiento en las tres morfologías descritas anteriormente. Seguidamente, se presentan los resultados de las pruebas de entrenamiento de las redes neuronales que se usaron para el algoritmo propuesto en esta investigación. Por último, se presentan los resultados de las pruebas de comparación de las dos técnicas de control de movimiento implementadas en la presente investigación.

7.1. Estructura robot EMERGE morfología cadena, tipo ELE y bloque

Las morfologías que se configuraron para la elaboración de los patrones de movimiento mediante el algoritmo de control basado en aprendizaje computacional fueron la morfología tipo cadena, tipo ELE y tipo bloque, estas morfologías se seleccionaron dado que la tipo cadena es la morfología básica que se puede replicar con el robot EMERGE; por otro lado, las morfologías tipo ELE y bloque han sido estudiadas por (Hernández Martínez et al., 2020) por lo cual es importante para la implementación de nuevos métodos de control de movimiento en estas morfologías.

En la Figura 29 se puede observar el ensamblaje final del robot EMERGE en la morfología bloque, implementando físicamente el diagrama de la Figura 20, así como en la Figura 30 se puede observar la configuración de los módulos con la respectiva identificación, allí se puede observar que el controlador local se encuentra sobre una pieza extra que está conectada al módulo 1, para permitir la libre rotación y movilidad del robot.

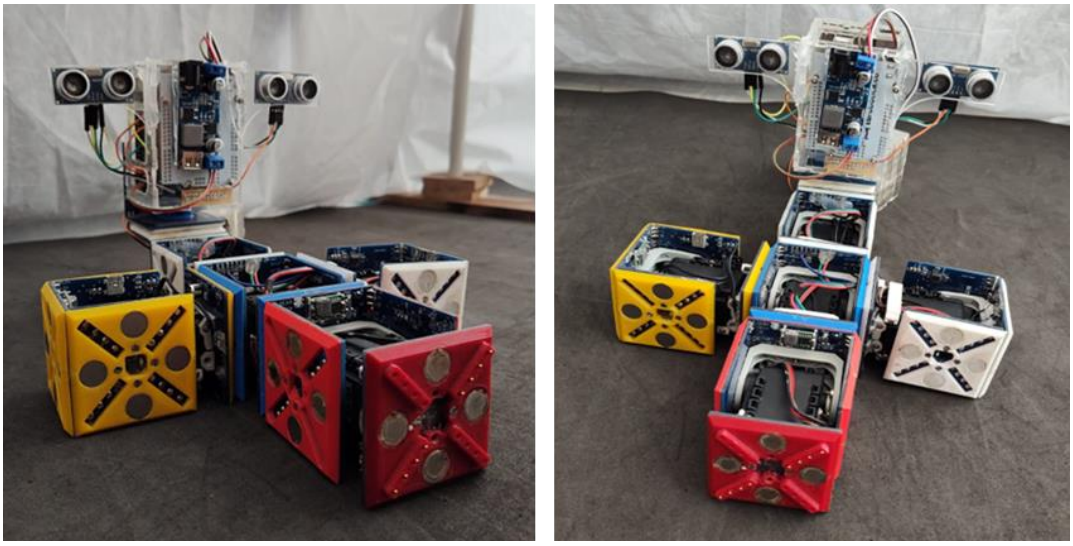


Figura 29. Ensamblaje robot EMERGE morfología bloque. Elaboración propia.

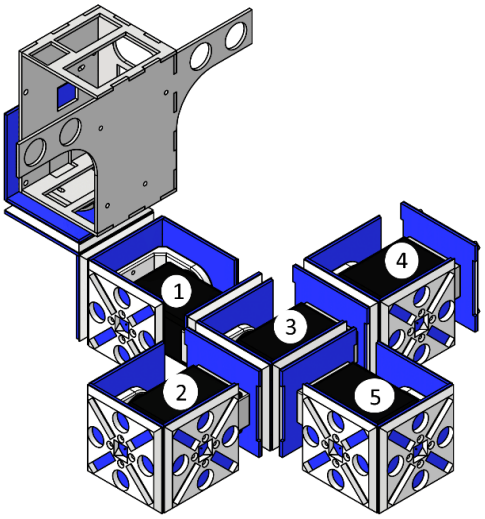


Figura 30. Configuración morfología bloque. Elaboración propia.

En la Figura 31 se puede observar el ensamblaje final del robot EMERGE en la morfología cadena, implementando físicamente el diagrama de la Figura 20, así como en la Figura 32 se puede observar la configuración de los módulos con la respectiva identificación, allí se puede observar que el controlador local se encuentra en el módulo central que en esta configuración es el identificado como 3, con el fin de generar estabilidad al robot en cualquier movimiento.

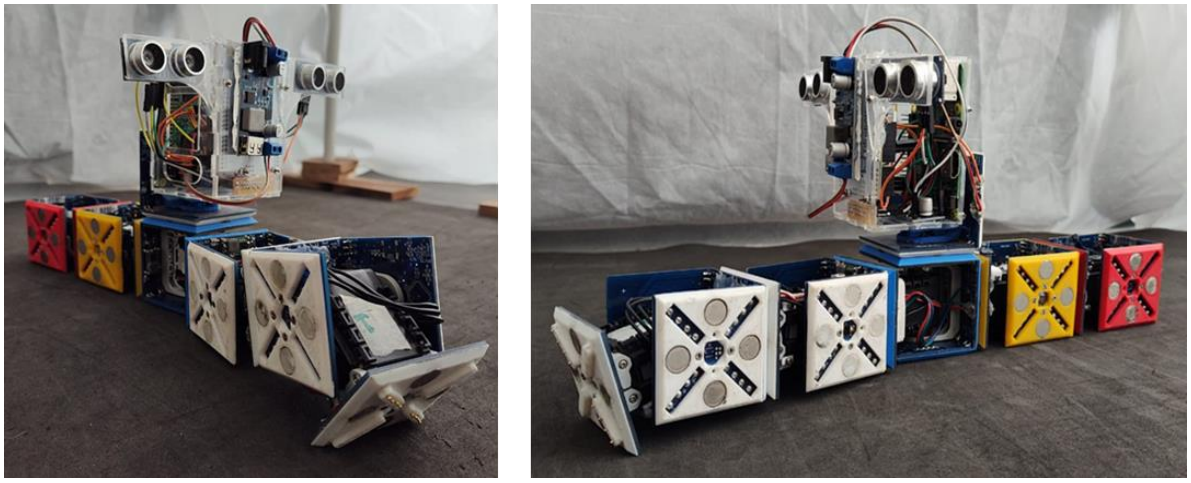


Figura 31. Ensamblaje robot EMERGE morfología cadena. Elaboración propia.

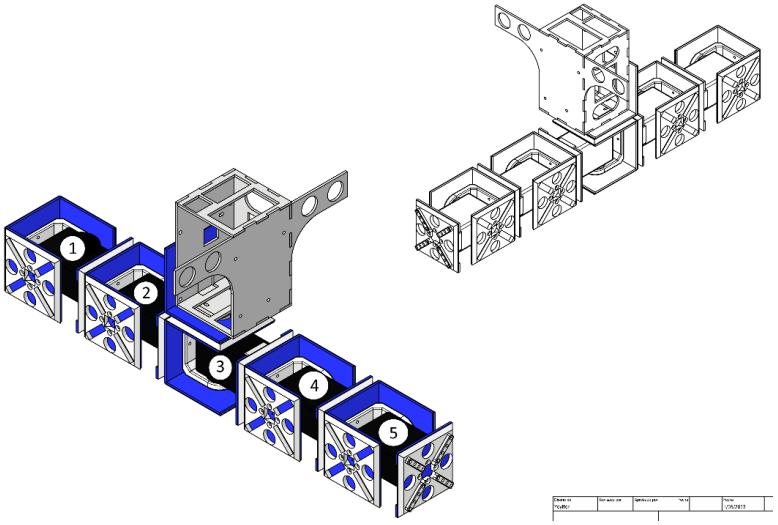


Figura 32. Configuración morfología cadena. Elaboración propia.

En la Figura 33 se puede observar el ensamblaje final del robot EMERGE en la morfología tipo ELE, implementando físicamente el diagrama de la Figura 20, así como en la Figura 34 se puede observar la configuración de los módulos con la respectiva identificación que forman una “L”, allí se puede observar que el controlador local se encuentra en el módulo cuatro, permitiendo la libertad de movimiento de los módulos 1, 2 y 3.

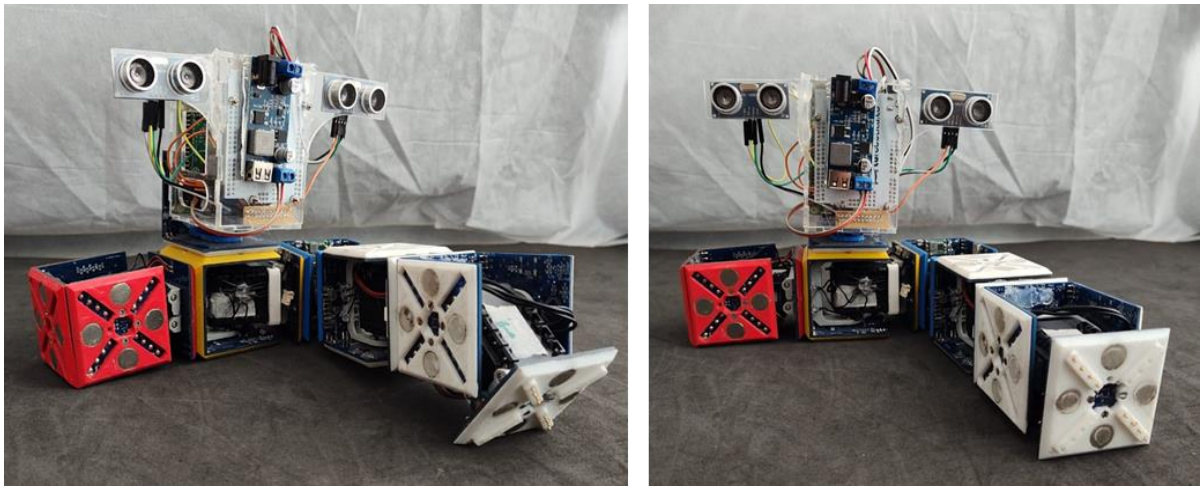


Figura 33. Ensamblaje robot EMERGE morfología ELE. Elaboración propia.

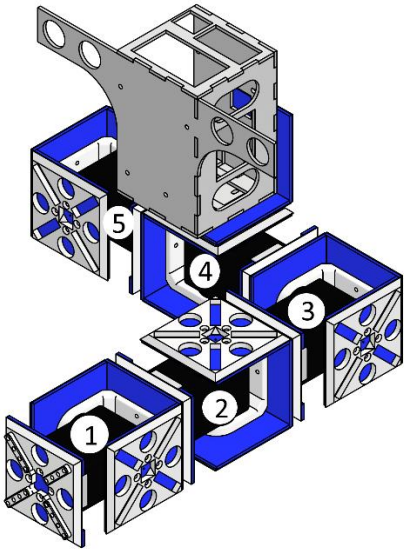


Figura 34. Configuración morfología tipo ELE. Elaboración propia.

7.2. Resultados de entrenamiento del robot con tablas de control de movimiento

En relación con el algoritmo de control de movimiento basado en tablas de movimiento desarrollado en la sección 6.2. Los resultados muestran que, en la morfología en cadena, los parámetros menos adecuados para el desplazamiento del robot son aquellos que tienen una velocidad del motor Dynamixel de 200 bits (equivalentes a 22.2 rpm) y una tasa de envío de trama de un paquete por segundo, tal como se muestra en la Figura 35. Se observó que, aunque se enviaron más paquetes de datos, como se puede apreciar en la línea gris, el robot no avanzó más de 40 cm, a diferencia de lo que sucedió con otros parámetros de velocidad y tiempo de envío de trama que permitieron un mejor desempeño del robot.

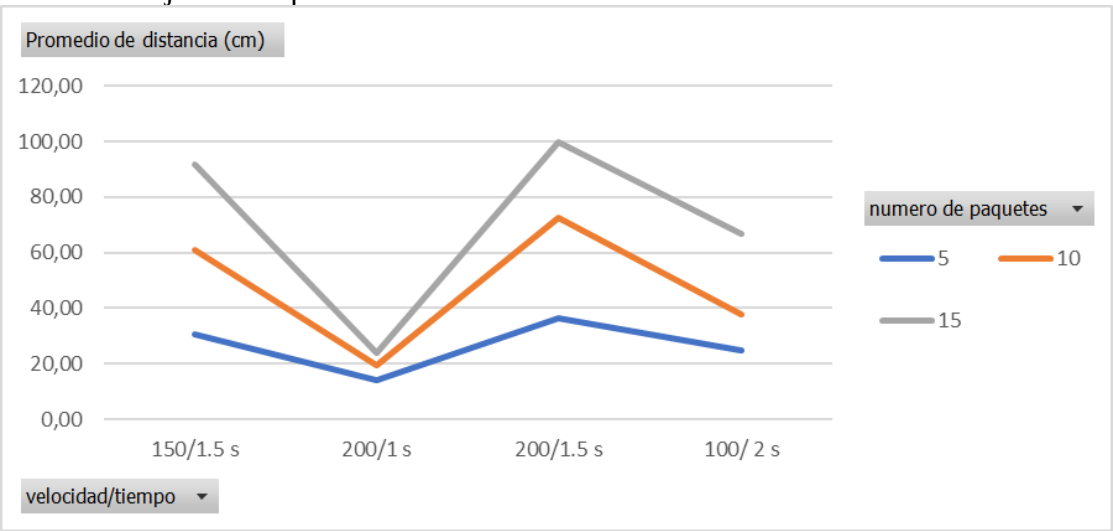


Figura 35. Gráfica de velocidad actuador (bits) sobre tiempo de envío trama (s) por distancia promedio (cm) en morfología cadena. Elaboración propia.

Por otro lado, al realizar el mismo experimento con la morfología tipo ELE, se observa que la variación de distancia no es significativa, aunque se varíen los parámetros de velocidad y tiempo de envío de trama. Solo se aprecia un cambio en el desempeño del robot cuando se envían la mayor cantidad de paquetes a una velocidad de motor baja (100 bits equivalentes a 11.1 rpm), en comparación con los otros tres parámetros, y con un tiempo de envío de trama alto, como se puede apreciar en la Figura 36.

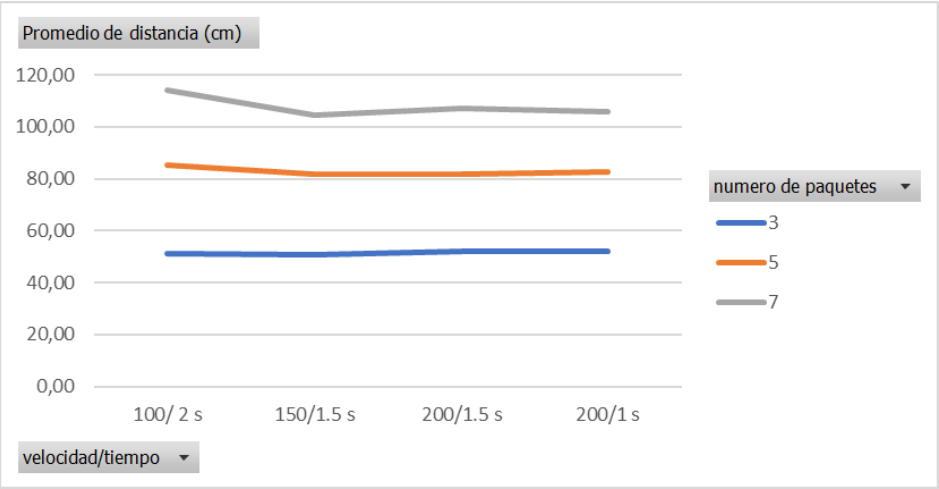


Figura 36. Gráfica de velocidad actuador (bits) sobre tiempo de envío trama (s) por distancia promedio (cm) en morfología ELE. Elaboración propia.

Finalmente, se llevó a cabo el experimento con la morfología en bloque. En esta ocasión, las distancias más largas alcanzadas con las distintas cantidades de paquetes se obtuvieron cuando se empleó una velocidad de actuador más alta (200 bits equivalentes a 22.2 rpm) y un tiempo de envío de trama bajo, como se aprecia en la Figura 37. Durante estas pruebas, se pudo notar que los actuadores con velocidades de 200 bits aumentan su temperatura.

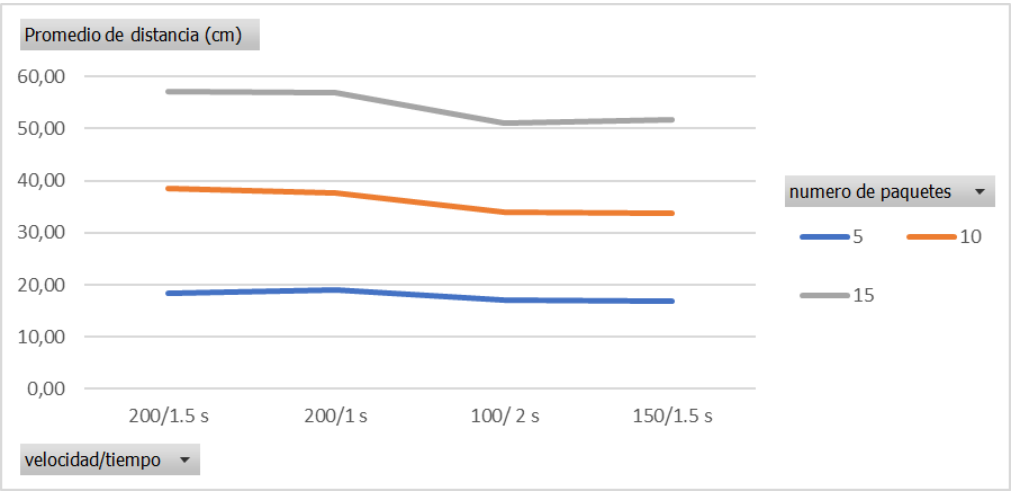


Figura 37. Gráfica de velocidad actuador (bits) sobre tiempo de envío trama (s) por distancia promedio (cm) en morfología bloque. Elaboración propia.

7.3. Entrenamiento de las redes neuronales

En la Tabla 2 se observan las diferentes configuraciones de parámetros como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología cadena, cuando no se detecta un obstáculo se obtiene que la mejor configuración es la del enfoque 4, que se refiere al descrito en la sección 6.2 que se entrenó en la Raspberry. Con este algoritmo se logró que, con tres capas, 1000 épocas y dos minutos de ejecución del entrenamiento, se lograra una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 2. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo cadena sensado de obstáculos en 0,0. Elaboración propia.

Morfología Cadena Avanzar (0,0)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	BinaryCrossentropy	0%	15
	15	1500	nadam	BinaryCrossentropy	1%	17
	20	1500	nadam	BinaryCrossentropy	2%	17
	20	1500	Adam	poisson	45%	18
	20	2000	Adam	poisson	60%	20
	30	3500	Adam	poisson	80%	35
	49	5000	Adam	poisson	99%	40
4	3	1000	nadam	MeanSquaredError	100%	2

En la Tabla 3 se observa el resumen de parámetros de entrenamiento, como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología cadena, cuando se sensa un obstáculo se obtiene que la mejor configuración es la del enfoque 4, que se refiere al descrito en la sección 6.2 que se entrenó en la Raspberry. Con este algoritmo se obtuvo que, con tres capas, 1000 épocas y tres minutos de ejecución del entrenamiento, una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 3. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo cadena sensado de obstáculos en 1, 0. Elaboración propia.

Morfología Cadena Girar (1,0)						
Enfoque	Cantidad de capas de redes	Épocas	Optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	BinaryCrossentropy	0%	16
	15	1600	nadam	BinaryCrossentropy	1%	18
	20	3000	nadam	BinaryCrossentropy	2%	20
	20	6500	Adam	poisson	45%	35
	20	5500	Adam	poisson	56%	37
	30	8000	Adam	poisson	75%	40

	52	15000	Adam	poisson	99%	55
4	3	1000	nadam	MeanSquaredError	100%	3

En la Tabla 4 se observan las diferentes configuraciones de parámetros como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología cadena, cuando se detecta un obstáculo se obtiene que la mejor configuración es la del enfoque 4, que se refiere al descrito en la sección 6.2 que se entrenó en la Raspberry. Con este algoritmo se logró que, con tres capas, 1000 épocas y dos minutos de ejecución del entrenamiento, se lograra una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 4. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo cadena sensado de obstáculos en 0, 1. Elaboración propia.

Morfología Cadena Girar (0,1)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	BinaryCrossentropy	0%	16
	15	1500	nadam	BinaryCrossentropy	2%	18
	20	3000	nadam	BinaryCrossentropy	2%	20
	20	4500	Adam	poisson	50%	35
	20	5500	Adam	poisson	60%	37
	30	8000	Adam	poisson	80%	40
	49	15000	Adam	poisson	99%	55
4	3	1000	nadam	MeanSquaredError	100%	3

En la Tabla 5 se presentan los diferentes de parámetros de entrenamiento de los modelos, como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología cadena cuando está presente un obstáculo, se obtiene que la mejor configuración es la del enfoque 4, que se refiere al descrito en la sección 6.2 entrenado en la Raspberry. Con este algoritmo se obtuvo con tres capas, 1000 épocas y tres minutos de ejecución del entrenamiento, una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 5. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo cadena sensado de obstáculos en 1,1. Elaboración propia.

Morfología Cadena Girar (1,1)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	BinaryCrossentropy	0%	16
	15	1500	nadam	BinaryCrossentropy	2%	18
	15	3000	nadam	BinaryCrossentropy	2%	22
	15	7500	Adam	poisson	40%	35

	20	10000	Adam	poisson	60%	40
	35	18000	Adam	poisson	76%	58
	55	23000	Adam	poisson	99%	75
4	3	1000	nadam	MeanSquaredError	100%	3

La Tabla 6 presenta las diferentes configuraciones aplicadas al entrenamiento del modelo, como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología bloque, cuando no se detecta un obstáculo se obtiene que la mejor configuración es la del enfoque 4, descrito en la sección 6.2. Con método de entrenamiento se logró que, con tres capas, 1000 épocas y tres minutos de ejecución del entrenamiento, se lograra una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 6. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo bloque sensado de obstáculos en 0,0. Elaboración propia.

Morfología Bloque Avanzar (0,0)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	CategoricalCrossentropy	0%	16
	15	1500	nadam	CategoricalCrossentropy	2%	22
	20	3000	nadam	CategoricalCrossentropy	1%	28
	20	3500	Adam	poisson	40%	35
	20	5500	Adam	poisson	60%	38
	30	5000	Adam	poisson	75%	47
	49	7000	Adam	poisson	99%	55
4	3	1000	nadam	MeanSquaredError	100%	3

En la Tabla 7 se presentan los diferentes parámetros de entrenamiento, como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los movimientos de la morfología bloque, cuando se detecta un obstáculo se obtiene que la mejor configuración es la del enfoque 4, detallada en la sección 6.2 Con este algoritmo se obtuvo un aprendizaje del 100% de los movimientos de la tabla de enseñanza con tres capas, 1000 épocas y cuatro minutos de ejecución del entrenamiento.

Tabla 7. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo bloque sensado de obstáculos en 1,0. Elaboración propia.

Morfología Bloque Girar (1,0)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	CategoricalCrossentropy	0%	16
	15	1500	nadam	CategoricalCrossentropy	2%	22
	20	3000	nadam	BinaryCrossentropy	2%	28

	20	3000	Adam	poisson	50%	35
	20	4500	Adam	poisson	68%	43
	35	6500	Adam	poisson	80%	53
	50	8500	Adam	poisson	99%	63
4	3	1000	nadam	MeanSquaredError	100%	4

La Tabla 8 presenta las diferentes configuraciones de parámetros de entrenamiento, como el número de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento en la morfología bloque, cuando se detecta un obstáculo se obtiene que la mejor configuración es la presentada en el enfoque 4, presentado en la sección 6.2. Con este algoritmo se obtuvo, con tres capas, 1000 épocas y cuatro minutos de ejecución del entrenamiento, una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 8. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo bloque sensado de obstáculos en 0,1. Elaboración propia.

Morfología Bloque Girar (0,1)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	MeanSquaredError	0%	16
	15	1500	nadam	MeanSquaredError	1%	22
	20	3000	nadam	MeanSquaredError	1%	28
	20	3500	Adam	poisson	50%	35
	20	3500	Adam	poisson	60%	43
	30	6500	Adam	poisson	83%	53
	55	9000	Adam	poisson	99%	63
4	3	1000	nadam	MeanSquaredError	100%	4

En la Tabla 9 se observan las diferentes configuraciones de parámetros para el entrenamiento del modelo correspondiente al escenario de un obstáculo sensado por los dos sensores, como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología bloque, se obtiene que la mejor configuración es la del enfoque 4. Con este algoritmo se logró que, con tres capas, 1000 épocas y dos minutos de ejecución del entrenamiento, obtener una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 9. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo bloque sensado de obstáculos en 1,1. Elaboración propia.

Morfología Bloque Girar (1,1)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	MeanSquaredError	0%	16

	15	1500	nadam	MeanSquaredError	1%	22
	20	1500	nadam	MeanSquaredError	1%	28
	24	3500	Adam	poisson	50%	35
	26	5000	Adam	poisson	60%	38
	32	7500	Adam	poisson	78%	47
	54	9000	Adam	poisson	99%	63
4	3	1000	nadam	MeanSquaredError	100%	4

En la Tabla 10 se presentan las configuraciones para el entrenamiento del modelo de movimiento, como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología tipo ELE, cuando no se detecta un obstáculo se obtiene que la mejor configuración es la del enfoque 4. Con este algoritmo se obtuvo que, con tres capas, 1000 épocas y tres minutos de ejecución del entrenamiento, se lograra una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 10. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo ELE sensado de obstáculos en 0,0. Elaboración propia.

Morfología tipo ELE Avanzar (0,0)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	MeanSquaredError	0%	16
	15	1500	nadam	poisson	8%	22
	20	3000	nadam	poisson	15%	28
	35	7000	Adam	poisson	35%	33
	38	13000	Adam	poisson	60%	36
	45	17000	Adam	poisson	83%	46
	55	20000	Adam	poisson	99%	49
4	3	1000	nadam	MeanSquaredError	100%	3

La Tabla 11 presenta los diferentes parámetros de entrenamiento como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología tipo ELE, cuando se detecta un obstáculo se obtiene que la mejor configuración es la del enfoque 4, que se refiere al descrito en la sección 6.2 que se entrenó en la Raspberry. Con esto se logró que, con tres capas, 1000 épocas y cuatro minutos de ejecución del entrenamiento, se lograra una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 11. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo ELE sensado de obstáculos en 1,0. Elaboración propia.

Morfología tipo ELE Girar (1,0)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	MeanSquaredError	0%	16
	15	1500	nadam	poisson	8%	22
	20	3400	nadam	poisson	16%	32
	35	7600	Adam	poisson	37%	43
	38	14000	Adam	poisson	60%	50
	45	16800	Adam	poisson	76%	66
	55	21000	Adam	poisson	99%	72
4	3	1000	nadam	MeanSquaredError	100%	4

En la Tabla 12 se presentan las principales configuraciones de parámetros para el entrenamiento del modelo, como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología tipo ELE, cuando se detecta un obstáculo se obtiene que la mejor configuración es la del enfoque 4, que se refiere al descrito en la sección 6.2 que se entrenó en la Raspberry. Con este algoritmo se logró que, con tres capas, 1000 épocas y cuatro minutos de ejecución del entrenamiento, se lograra una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red.

Tabla 12. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo ELE sensado de obstáculos en 0,1. Elaboración propia.

Morfología tipo ELE Girar (0,1)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	MeanSquaredError	0%	16
	15	1500	nadam	poisson	8%	22
	20	3000	nadam	poisson	15%	28
	35	7000	Adam	poisson	38%	43
	38	13400	Adam	poisson	55%	50
	45	16000	Adam	poisson	79%	63
	55	20500	Adam	poisson	99%	76
4	3	1000	nadam	MeanSquaredError	100%	4

En la Tabla 13 se presentan las diferentes configuraciones de parámetros como la cantidad de capas ocultas, cantidad de épocas, tipo de optimizador, tipo de perdida y la precisión del entrenamiento. Para los patrones de movimiento de la morfología tipo ELE, cuando se detecta un obstáculo se obtiene que la mejor configuración es la del enfoque 4, que se refiere al descrito en la sección 6.2 que se entrenó en la Raspberry. Con este algoritmo se logró que, con tres capas, 1000 épocas y

cuatro minutos de ejecución del entrenamiento, se lograra una precisión del 100% que equivale a un aprendizaje total de la tabla de movimiento enseñada a la red, con este enfoque se logró una reducción de 74 minutos en el entrenamiento del modelo de este movimiento.

Tabla 13. Entrenamiento de las redes neuronales con distintas configuraciones de entrenamiento para morfología tipo ELE sensado de obstáculos en 1,1. Elaboración propia.

Morfología tipo ELE Girar (1,1)						
Enfoque	Cantidad de capas de redes	Épocas	Tipo optimizador	Tipo de pérdida	Precisión	Tiempo de ejecución (min)
2	10	1000	nadam	MeanSquaredError	0%	16
	15	1500	nadam	poisson	8%	22
	26	4600	nadam	poisson	26%	35
	44	8000	Adam	poisson	48%	46
	36	15000	Adam	poisson	60%	55
	44	17500	Adam	poisson	80%	60
	57	22000	Adam	poisson	99%	78
4	3	1000	nadam	MeanSquaredError	100%	4

7.4. Comparación para el control de movimiento del robot EMERGE con tablas de movimiento y redes neuronales en entornos llanos y con obstáculos

La elaboración de las pruebas para la comparación de modelos de control de movimiento en la morfología cadena se creó un entorno como se ve en la Figura 38 y Figura 39, allí se ven tres medidas con respecto al punto inicial del robot, estas medidas hacen referencia a las distintas distancias de referencia que se tomaron para realizar las pruebas de funcionamiento. La primera delimitación de derecha a izquierda es la línea de los 50cm, la segunda hace referencia a la línea de los 75cm y por último se encuentra la de los 100cm. Adicionalmente, en la Figura 38 se puede observar un cubo, el cual es la representación del obstáculo, que fue ubicado a una distancia de 50cm, 75cm y 100cm.

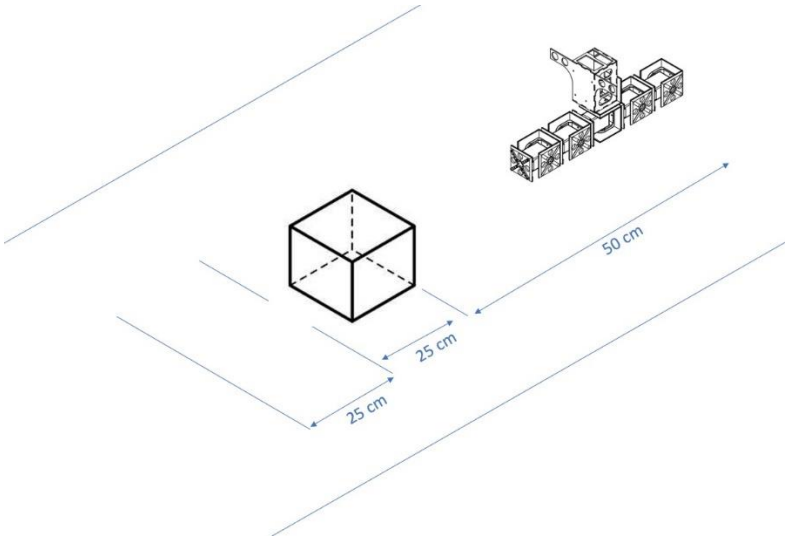


Figura 38. Plano de visualización del entorno de pruebas para la morfología cadena con evasión de obstáculo. Elaboración propia.

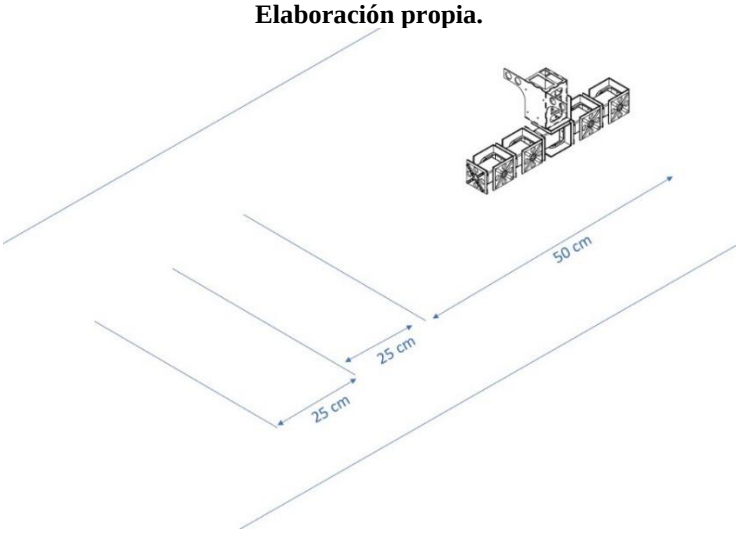


Figura 39. Plano de visualización del entorno de pruebas para la morfología cadena en entorno llano. Elaboración propia.

En la Tabla 14 y Tabla 15 se resumen los resultados de las cinco pruebas realizadas para la morfología cadena para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 50cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la derecha del robot.

En la Figura 40, se observa que en promedio el tiempo de desplazamiento que toma el control de movimiento por tablas de movimiento es de 57.4 segundos, mientras que el promedio por control RNA es de 48.4 segundos, esto es 9 segundos menos que lo obtenido por tablas de movimiento en promedio. Igualmente se evidencia que bajo el control de movimiento por tablas de movimiento la

repetitividad de la distancia promedio de evasión con respecto al obstáculo es menor que con el control con RNA.

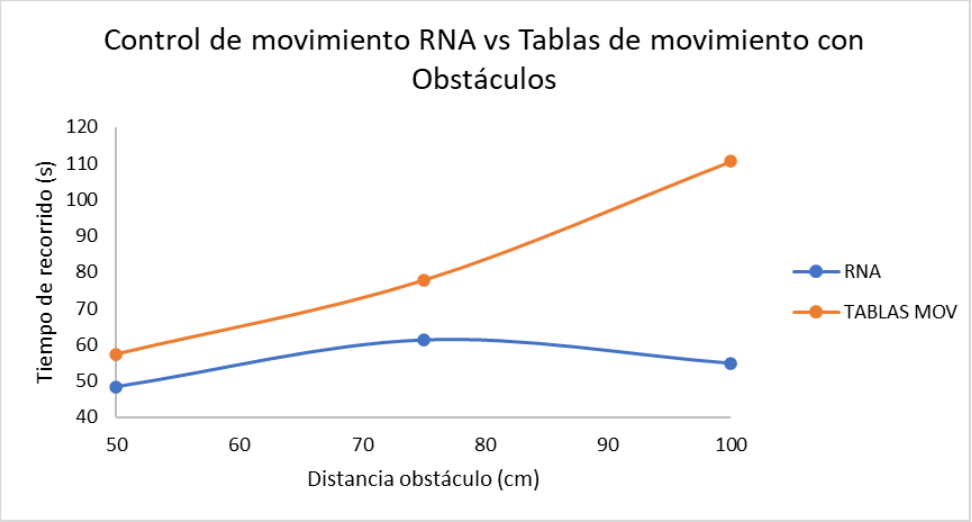


Figura 40. Representación gráfica morfología cadena experimento con obstáculos. Elaboración propia.

Tabla 14. Resultados experimento morfología cadena con obstáculo a 50cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos en morfología cadena (TABLA DE MOVIMIENTO)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
57	50	SI	24	Izquierda	Izquierda
58	50	SI	25	Izquierda	Izquierda
57	50	SI	25	Izquierda	Izquierda
58	50	SI	24	Izquierda	Izquierda
57	50	SI	25	Izquierda	Izquierda

Tabla 15. Resultados experimento morfología cadena con obstáculo a 50cm en la derecha, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología cadena (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
48	50	SI	25	Izquierda	Izquierda
49	50	SI	24	Izquierda	Izquierda
49	50	SI	25	Izquierda	Izquierda
48	50	SI	25	Izquierda	Izquierda
48	50	SI	25	Izquierda	Izquierda

En la Tabla 16 y Tabla 17 se resumen los resultados de las cinco pruebas realizadas para la morfología cadena para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó el obstáculo a 75cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si hubo detección del obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la derecha del robot.

En promedio el tiempo de desplazamiento con el control de movimiento por tablas de movimiento es de 77.8 segundos, mientras que el promedio por control RNA es de 61.4 segundos, esto es 16.4 segundos menos que lo obtenido por tablas de movimiento en promedio, como se ve en la Figura 40. Igualmente se evidencia que bajo el control de movimiento por tablas de movimiento la repetitividad de la distancia promedio de evasión con respecto al obstáculo es menor que con el control con RNA.

Tabla 16. Resultados experimento morfología cadena con obstáculo a 75cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos en morfología cadena (TABLA DE MOVIMIENTO)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
77	75	SI	24	Izquierda	Izquierda
78	75	SI	25	Izquierda	Izquierda
77	75	SI	25	Izquierda	Izquierda
79	75	SI	24	Izquierda	Izquierda
78	75	SI	22	Izquierda	Izquierda

Tabla 17. Resultados experimento morfología cadena con obstáculo a 75cm en la derecha, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología cadena (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
61	75	SI	22	Izquierda	Izquierda
62	75	SI	22	Izquierda	Izquierda
62	75	SI	22	Izquierda	Izquierda
61	75	SI	23	Izquierda	Izquierda
61	75	SI	22	Izquierda	Izquierda

En la Tabla 18 y Tabla 19 se resumen los resultados de las cinco pruebas realizadas para la morfología cadena para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó el obstáculo a 100cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el

obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la derecha del robot.

En la Figura 40 se observa que el tiempo de desplazamiento en promedio el control de movimiento por tablas de movimiento es de 110.6 segundos, mientras que el promedio por control RNA es de 92.6 segundos, esto es 18 segundos menos que lo obtenido por tablas de movimiento en promedio. Igualmente se evidencia que por medio del control de movimiento por tablas de movimiento la repetitividad de la distancia promedio de evasión con respecto al obstáculo es menor que con el control con RNA.

Tabla 18. Resultados experimento morfología cadena con obstáculo a 100cm en la derecha, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología cadena (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
93	100	SI	23	Izquierda	Izquierda
92	100	SI	23	Izquierda	Izquierda
92	100	SI	22	Izquierda	Izquierda
93	100	SI	23	Izquierda	Izquierda
93	100	SI	23	Izquierda	Izquierda

Tabla 19. Resultados experimento morfología cadena con obstáculo a 100cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos en morfología cadena (TABLA DE MOVIMIENTO)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
110	100	SI	24	Izquierda	Izquierda
110	100	SI	23	Izquierda	Izquierda
110	100	SI	24	Izquierda	Izquierda
111	100	SI	23	Izquierda	Izquierda
112	100	SI	24	Izquierda	Izquierda

En la Tabla 20 y Tabla 21 se resumen los resultados de las cinco pruebas realizadas para la morfología cadena para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 50cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la izquierda del robot.

El tiempo de desplazamiento promedio en el control de movimiento por tablas de movimiento es de 58.2 segundos, mientras que el promedio por control RNA es de 48.4 segundos, esto es 9.8 segundos menos que lo obtenido por tablas de movimiento en promedio. Igualmente se evidencia que bajo el control de movimiento por tablas de movimiento la repetitividad de la distancia promedio de evasión con respecto al obstáculo es menor que con el control con RNA.

Tabla 20. Resultados experimento morfología cadena con obstáculo a 50cm en la izquierda, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología cadena (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
48	50	SI	25	Derecha	Derecha
48	50	SI	24	Derecha	Derecha
48	50	SI	25	Derecha	Derecha
49	50	SI	25	Derecha	Derecha
49	50	SI	25	Derecha	Derecha

Tabla 21. Resultados experimento morfología cadena con obstáculo a 50cm en la izquierda, con control de movimiento por tabla de movimiento. Elaboración propia.

Detección de obstáculos en morfología cadena (TABLA DE MOVIMIENTO)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
58	50	SI	25	Derecha	Derecha
58	50	SI	24	Derecha	Derecha
59	50	SI	25	Derecha	Derecha
58	50	SI	25	Derecha	Derecha
58	50	SI	24	Derecha	Derecha

En la Tabla 22 y Tabla 23 se resumen los resultados de las cinco pruebas realizadas para la morfología cadena para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 75cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la izquierda del robot.

Se observa que el tiempo de desplazamiento promedio en el control de movimiento por tablas de movimiento es de 77.8 segundos, mientras que el promedio por control RNA es de 61.4 segundos, esto es 16.4 segundos menos que lo obtenido por tablas de movimiento en promedio. Igualmente, se evidencia que con el control de movimiento por tablas de movimiento la repetitividad de la distancia promedio de evasión con respecto al obstáculo es menor que con el control con RNA.

Tabla 22. Resultados experimento morfología cadena con obstáculo a 75cm en la izquierda, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología cadena (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
62	75	SI	22	Derecha	Derecha
61	75	SI	22	Derecha	Derecha
61	75	SI	22	Derecha	Derecha
62	75	SI	23	Derecha	Derecha
61	75	SI	22	Derecha	Derecha

Tabla 23. Resultados experimento morfología cadena con obstáculo a 75cm en la izquierda, con control de movimiento por tabla de movimiento. Elaboración propia.

Detección de obstáculos en morfología cadena (TABLA DE MOVIMIENTO)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
78	75	SI	22	Derecha	Derecha
77	75	SI	22	Derecha	Derecha
77	75	SI	24	Derecha	Derecha
78	75	SI	24	Derecha	Derecha
79	75	SI	22	Derecha	Derecha

En la Tabla 24 y Tabla 25 se resumen los resultados de las cinco pruebas realizadas para la morfología cadena para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó el obstáculo a 100cm del robot y se tomaron como parámetros comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la izquierda del robot.

Se observa que en promedio el tiempo de desplazamiento con el control de movimiento por tablas de movimiento es de 109.8 segundos, mientras que el promedio por control RNA es de 92.4 segundos, esto es 17.4 segundos menos que lo obtenido por tablas de movimiento en promedio. Igualmente se evidencia que por medio el control de movimiento por tablas de movimiento la repetitividad de la distancia promedio de evasión con respecto al obstáculo es menor que con el control con RNA.

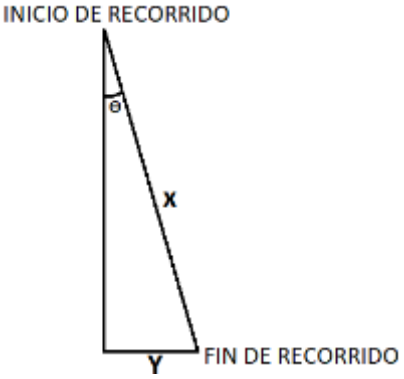
Tabla 24. Resultados experimento morfología cadena con obstáculo a 100cm en la izquierda, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología cadena (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
92	100	SI	23	Derecha	Derecha
93	100	SI	23	Derecha	Derecha
92	100	SI	22	Derecha	Derecha
93	100	SI	23	Derecha	Derecha
92	100	SI	23	Derecha	Derecha

Tabla 25. Resultados experimento morfología cadena con obstáculo a 100cm en la izquierda, con control de movimiento por tabla de movimiento. Elaboración propia.

Detección de obstáculos en morfología cadena (TABLA DE MOVIMIENTO)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
109	100	SI	23	Derecha	Derecha
110	100	SI	23	Derecha	Derecha
111	100	SI	24	Derecha	Derecha
109	100	SI	23	Derecha	Derecha
110	100	SI	22	Derecha	Derecha

En la Tabla 26 y Tabla 27 se resumen los resultados de las cinco pruebas realizadas para la morfología cadena para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se hicieron tres delimitaciones, la primera a 50cm, la segunda a 75 cm y la última a 100cm del robot; se tomaron como mediciones de comparación el tiempo para llegar a los puntos mencionados anteriormente, la distancia en Y que equivale a la diferencia entre el punto inicial y el punto final sobre el eje Y, con estos datos y la distancia delimitada para cada prueba se calculó el ángulo de desfase con el teorema de Pitágoras.



En la Figura 41 se observa que el tiempo de desplazamiento hasta la distancia objetivo en promedio con el control de movimiento por tablas de movimiento es de 51.2 segundos para una distancia de 50cm, 71.83 segundos para una distancia de 75cm y 99.83 segundos para una distancia de 100cm, mientras que el promedio por control RNA es de 46.6 segundos para una distancia de 50cm, 67.83 segundos para una distancia de 75cm y 99.83 para una distancia de 100cm, esto es 4.6 segundos menos para 50cm, 4 segundos menos para 70 cm menos que lo obtenido por tablas de movimiento. Sin embargo, para la distancia de 100cm el comportamiento del tiempo promedio es el mismo.

Igualmente, se evidencia en la Figura 42 que bajo el control de movimiento por tablas de movimiento el promedio del ángulo de desfase para 50cm fue de 5,48°, para 75cm fue de 5,71° y para 100cm fue de 7,69°; en contra parte, con el ángulo de desfase con el control de movimiento de la RNA fue de 2,63° para una distancia de 50cm, 4,64° para una distancia de 75cm y de 5,90° para una distancia de 100cm. Con lo cual se puede ver que con la RNA se disminuyó el promedio de desfase del robot permitiendo que el desplazamiento sea más recto.

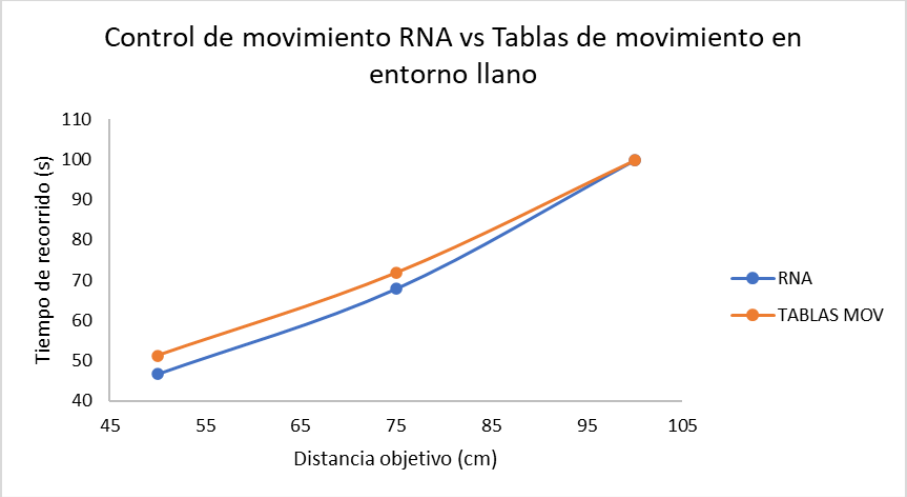


Figura 41. Representación gráfica morfología cadena experimento en entorno llano. Elaboración propia.

Tabla 26. Resultados experimento morfología cadena en entorno llano, con control de movimiento por tabla de movimiento. Elaboración propia.

MORFOLOGÍA CADENA RECORRIDO SIN OBSTACULOS (TABLA DE MOVIMIENTOS)			
Tiempo (s)	**Distancia H (cm)**	**Distancia Y (cm)**	**Angulo de desfase °**
52	50	5	5,71
51	50	5	5,71
51	50	4	4,57
52	50	5	5,71
50	50	5	5,71
71	75	7	5,33
72	75	7	5,33
73	75	8	6,09
73	75	7	5,33

72	75	8	6,09
70	75	8	6,09
101	100	13	7,41
102	100	14	7,97
102	100	14	7,97
95	100	13	7,41
101	100	13	7,41
98	100	14	7,97

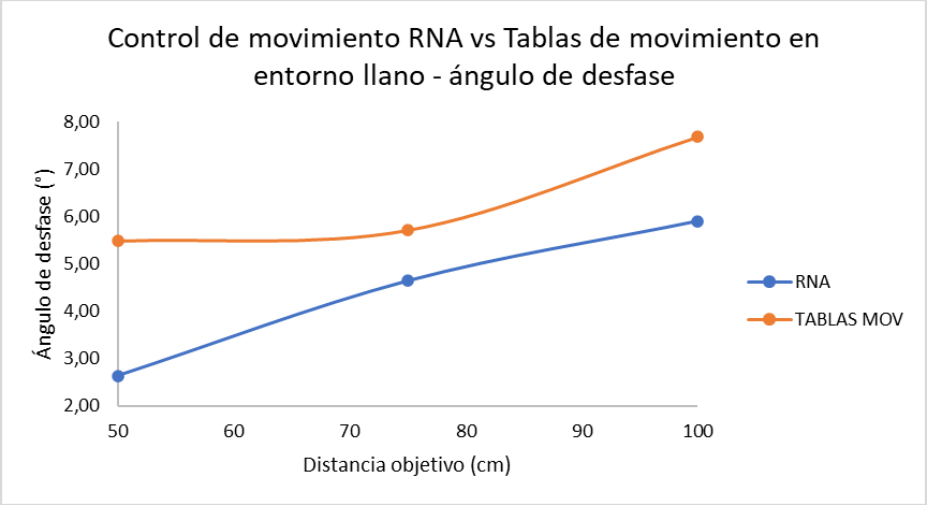


Figura 42. Representación gráfica ángulo de desfase morfología cadena experimento en entorno llano. Elaboración propia.

Tabla 27. Resultados experimento morfología cadena en entorno llano, con control de movimiento por RNA. Elaboración propia.

MORFOLOGÍA CADENA RECORRIDO SIN OBSTACULOS (RNA)			
Tiempo (s)	**Distancia H (cm)**	**Distancia Y (cm)**	**Ángulo de desfase °**
45	50	3	3,43
46	50	2	2,29
48	50	2,5	2,86
47	50	2	2,29
47	50	2	2,29
68	75	6	4,57
69	75	5	3,81
67	75	7	5,33
68	75	6,5	4,95
67	75	5	3,81
68	75	7	5,33
101	100	11	6,28

102	100	12	6,84
102	100	8	4,57
95	100	9	5,14
101	100	12	6,84
98	100	10	5,71

La elaboración de las pruebas para la comparación de modelos de control de movimiento en la morfología bloque se creó un entorno como se ve en la Figura 43 y Figura 44, allí se ven tres medidas con respecto al punto inicial del robot, estas medidas hacen referencia a las distintas distancias de referencia que se tomaron para realizar las pruebas de funcionamiento. La primera delimitación de derecha a izquierda es la línea de los 50cm, la segunda hace referencia a la línea de los 60cm y por último se encuentra la de los 75cm. Adicionalmente, en la Figura 38 se observa un cubo el cual es la representación del obstáculo, que fue ubicado a una distancia de 50cm, 60cm y 75cm.

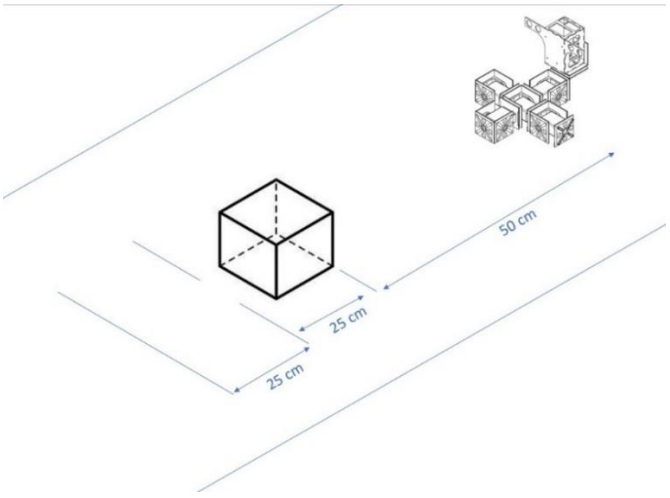


Figura 43. Plano de visualización del entorno de pruebas para la morfología bloque con evasión de obstáculo. Elaboración propia.

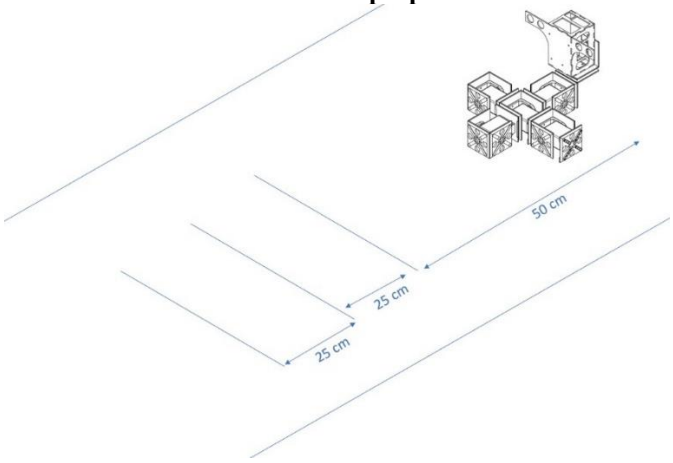


Figura 44. Plano de visualización del entorno de pruebas para la morfología bloque en entorno llano. Elaboración propia.

En la Tabla 28 y Tabla 29 se resumen los resultados de las cinco pruebas realizadas para la morfología bloque para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 50cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la derecha del robot.

Se observa que el tiempo de desplazamiento hasta la evasión del obstáculo en promedio con el control de movimiento por tablas de movimiento es de 158 segundos, mientras que el promedio por control RNA es de 150.6 segundos, esto es 7.4 segundos menos que lo obtenido por tablas de movimiento como se ve en la Figura 45.

Tabla 28. Resultados experimento morfología bloque con obstáculo a 50cm en la derecha, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología bloque (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
150	50	SI	10	Izquierda	Izquierda
151	50	SI	10	Izquierda	Izquierda
151	50	SI	9	Izquierda	Izquierda
150	50	SI	10	Izquierda	Izquierda
151	50	SI	10	Izquierda	Izquierda

Tabla 29. Resultados experimento morfología bloque con obstáculo a 50cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos en morfología bloque (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
157	50	SI	11	Izquierda	Izquierda
158	50	SI	10	Izquierda	Izquierda
160	50	SI	11	Izquierda	Izquierda
157	50	SI	11	Izquierda	Izquierda
158	50	SI	10	Izquierda	Izquierda

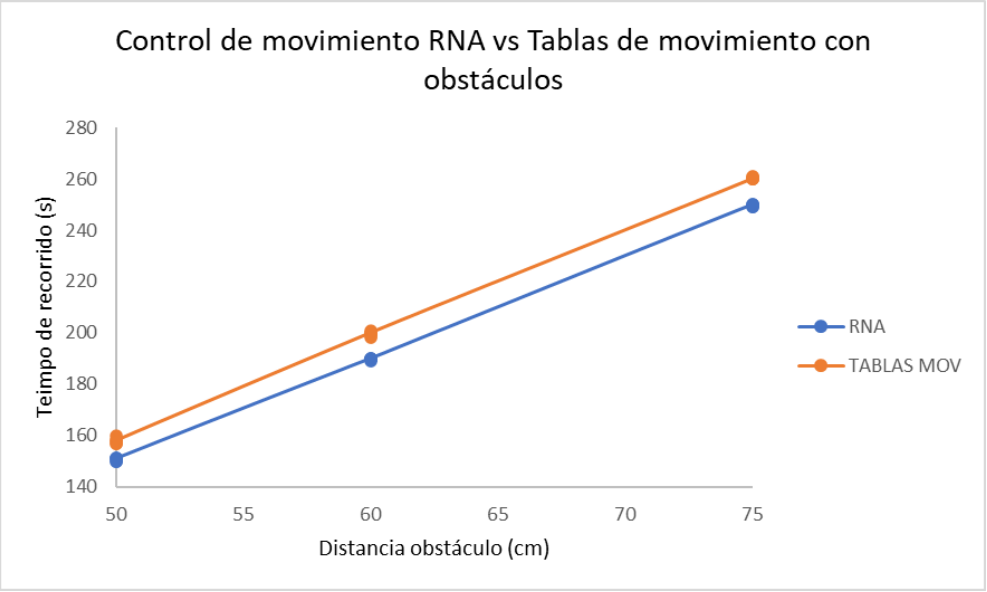


Figura 45. Representación gráfica morfología bloque experimento con obstáculos. Elaboración propia.

En la Tabla 30 y Tabla 31 se resumen los resultados de las cinco pruebas realizadas para la morfología bloque para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 60cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la derecha del robot.

El tiempo de desplazamiento hasta la evasión del obstáculo en promedio en el control de movimiento por tablas de movimiento es de 199.6 segundos, mientras que el promedio por control RNA es de 189.8 segundos, esto es 9.8 segundos menos que lo obtenido por tablas de movimiento como se ve en la Figura 45.

Tabla 30. Resultados experimento morfología bloque con obstáculo a 60cm en la derecha, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología bloque (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
190	60	SI	9	Izquierda	Izquierda
189	60	SI	10	Izquierda	Izquierda
190	60	SI	10	Izquierda	Izquierda
190	60	SI	9	Izquierda	Izquierda
190	60	SI	10	Izquierda	Izquierda

Tabla 31. Resultados experimento morfología bloque con obstáculo a 60cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos en morfología bloque (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
200	60	SI	9	Izquierda	Izquierda
199	60	SI	11	Izquierda	Izquierda
201	60	SI	10	Izquierda	Izquierda
198	60	SI	10	Izquierda	Izquierda
200	60	SI	11	Izquierda	Izquierda

En la Tabla 32 y Tabla 33 se resumen los resultados de las cinco pruebas realizadas para la morfología bloque para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 75cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la derecha del robot.

El tiempo de desplazamiento hasta la evasión del obstáculo en promedio con control de movimiento por tablas de movimiento es de 260.4 segundos, mientras que el promedio por control RNA es de 249.6 segundos como se observa en el gráfico de la Figura 45, esto es 10.8 segundos menos que lo obtenido por tablas de movimiento.

Tabla 32. Resultados experimento morfología bloque con obstáculo a 75cm en la derecha, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología bloque (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
250	75	SI	10	Izquierda	Izquierda
249	75	SI	11	Izquierda	Izquierda
249	75	SI	9	Izquierda	Izquierda
250	75	SI	10	Izquierda	Izquierda
250	75	SI	10	Izquierda	Izquierda

Tabla 33. Resultados experimento morfología bloque con obstáculo a 75cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos en morfología bloque (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
260	75	SI	10	Izquierda	Izquierda

261	75	SI	11	Izquierda	Izquierda
260	75	SI	9	Izquierda	Izquierda
261	75	SI	10	Izquierda	Izquierda
260	75	SI	11	Izquierda	Izquierda

En la Tabla 34 y Tabla 35 se resumen los resultados de las cinco pruebas realizadas para la morfología bloque para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 50cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la izquierda del robot.

Se observa el tiempo de desplazamiento hasta la evasión del obstáculo en promedio con el control de movimiento por tablas de movimiento es de 157.4 segundos, mientras que el promedio por control RNA es de 150.6 segundos, esto es 7.1 segundos menos que lo obtenido por tablas de movimiento.

Tabla 34. Resultados experimento morfología bloque con obstáculo a 50cm en la izquierda, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología bloque (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
150	50	SI	10	Derecha	Derecha
151	50	SI	10	Derecha	Derecha
151	50	SI	9	Derecha	Derecha
150	50	SI	10	Derecha	Derecha
151	50	SI	10	Derecha	Derecha

Tabla 35. Resultados experimento morfología bloque con obstáculo a 50cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos en morfología bloque (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
158	50	SI	10	Derecha	Derecha
157	50	SI	10	Derecha	Derecha
157	50	SI	11	Derecha	Derecha
158	50	SI	10	Derecha	Derecha
157	50	SI	10	Derecha	Derecha

En la Tabla 36 y Tabla 37 se resumen los resultados de las cinco pruebas realizadas para la morfología bloque para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 60cm del robot y se tomaron medidas de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la izquierda del robot.

El tiempo promedio de desplazamiento hasta la evasión del obstáculo en el control de movimiento por tablas de movimiento es de 200.4 segundos, mientras que el promedio por control RNA es de 189.8 segundos, esto es 10.6 segundos menos que lo obtenido por tablas de movimiento.

Tabla 36. Resultados experimento morfología bloque con obstáculo a 60cm en la izquierda, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología bloque (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
190	60	SI	9	Derecha	Derecha
189	60	SI	10	Derecha	Derecha
190	60	SI	10	Derecha	Derecha
190	60	SI	9	Derecha	Derecha
190	60	SI	10	Derecha	Derecha

Tabla 37. Resultados experimento morfología bloque con obstáculo a 60cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos en morfología bloque (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
201	60	SI	9	Derecha	Derecha
201	60	SI	11	Derecha	Derecha
199	60	SI	10	Derecha	Derecha
200	60	SI	9	Derecha	Derecha
201	60	SI	10	Derecha	Derecha

En la Tabla 38 y Tabla 39 se resumen los resultados de las cinco pruebas realizadas para la morfología bloque para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 75cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la izquierda del robot.

Se observar que el tiempo de desplazamiento hasta la evasión del obstáculo en promedio con el control de movimiento por tablas de movimiento es de 220.6 segundos, mientras que el promedio

por control RNA es de 249.6 segundos esto es 29 segundos más que lo obtenido por tablas de movimiento.

Tabla 38. Resultados experimento morfología bloque con obstáculo a 75cm en la izquierda, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología bloque (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
250	75	SI	10	Derecha	Derecha
249	75	SI	11	Derecha	Derecha
249	75	SI	9	Derecha	Derecha
250	75	SI	10	Derecha	Derecha
250	75	SI	10	Derecha	Derecha

Tabla 39. Resultados experimento morfología bloque con obstáculo a 75cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos en morfología bloque (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
259	75	SI	11	Derecha	Derecha
261	75	SI	11	Derecha	Derecha
162	75	SI	10	Derecha	Derecha
161	75	SI	10	Derecha	Derecha
260	75	SI	11	Derecha	Derecha

En la Tabla 26 y Tabla 27 se resumen los resultados de las cinco pruebas realizadas para la morfología bloque para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se hicieron tres delimitaciones, la primera a 50cm, la segunda a 60 cm y la última a 75cm del robot y se tomaron las mismas mediciones que con la morfología cadena.

El tiempo de desplazamiento a la distancia objetivo en promedio con el control de movimiento por tablas de movimiento es de 191 segundos para una distancia de 50cm, 231,5 segundos para una distancia de 60cm y 292.5 segundos para una distancia de 75cm; en cambio, el promedio por control RNA es de 182.6 segundos para una distancia de 50cm, 218.33 segundos para una distancia de 60cm y 278.83 para una distancia de 75cm, con esto se obtiene que para la morfología bloque el rendimiento de la RNA es mejor que con las tablas de movimiento, ya que en las tres distancias el tiempo de desplazamiento fue mayor como se ve en la Figura 46.

Igualmente, se observa que bajo el control de movimiento por tablas de movimiento el promedio del ángulo de desfase para 50cm fue de 3,18°, para 60cm fue de 3,40° y para 75cm fue de 4,09°; en contra parte, con el ángulo de desfase con el control de movimiento de la RNA fue de 0,64° para

una distancia de 50cm, 0,76° para una distancia de 60cm y de 0,81° para una distancia de 75cm. Con lo cual se ve que con la RNA se disminuyó el promedio de desfase del robot permitiendo que el desplazamiento sea más recto como se observa en la Figura 47.

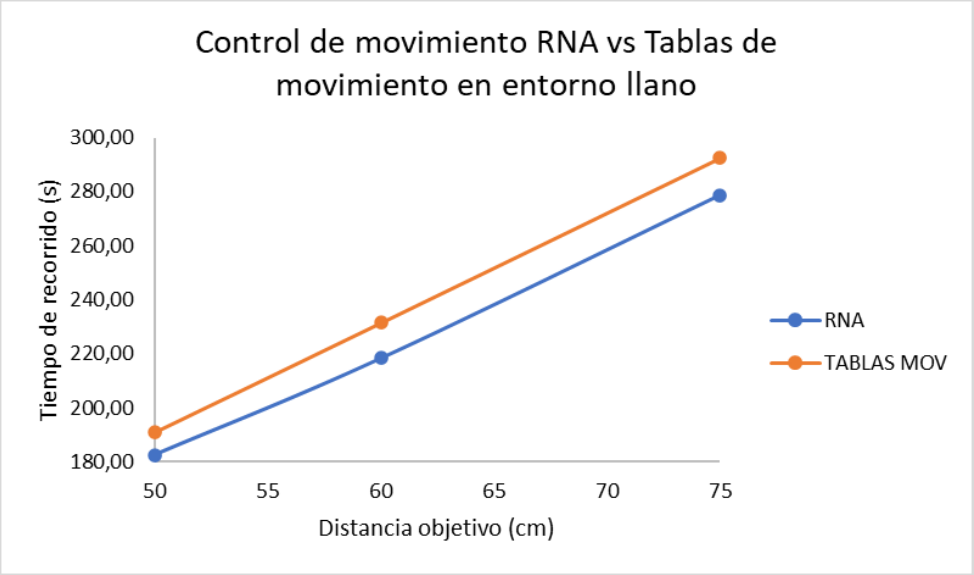


Figura 46. Representación gráfica morfología bloque experimento en entorno llano. Elaboración propia.

Tabla 40. Resultados experimento morfología bloque en entorno llano, con control de movimiento por RNA. Elaboración propia.

Morfología bloque recorrido sin obstáculos (RNA)			
Tiempo (s)	**Distancia H (cm)**	**Distancia Y (cm)**	**Angulo de desfase °**
183	50	0,5	0,57
182	50	0,6	0,69
183	50	0,4	0,46
183	50	0,7	0,80
182	50	0,6	0,69
218	60	0,8	0,76
218	60	0,8	0,76
219	60	0,9	0,86
218	60	0,8	0,76
218	60	0,7	0,67
219	60	0,8	0,76
280	75	0,9	0,69
279	75	1	0,76
278	75	1,3	0,99
280	75	1	0,76
278	75	1	0,76
278	75	1,2	0,92

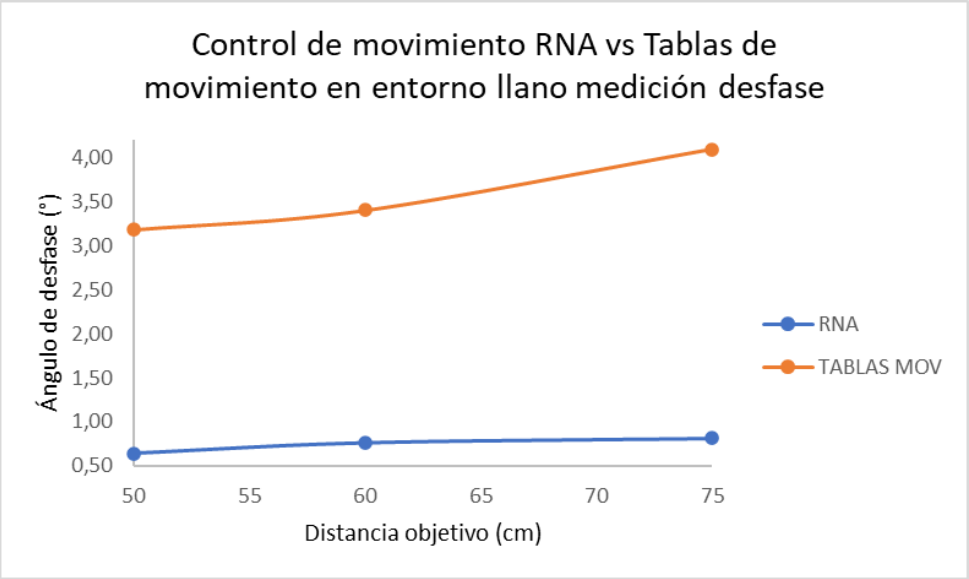


Figura 47. Representación gráfica morfología bloque experimento en entorno llano medición ángulo de desfase. Elaboración propia.

Tabla 41. Resultados experimento morfología bloque en entorno llano, con control de movimiento por tablas de movimiento. Elaboración propia.

Morfología bloque recorrido sin obstáculos (tablas de movimiento)			
Tiempo (s)	**Distancia H (cm)**	**Distancia Y (cm)**	**Angulo de desfase °**
190	50	3	3,43
191	50	2,6	2,98
190	50	2,8	3,21
192	50	2,7	3,09
192	50	2,8	3,21
230	60	3	2,86
232	60	3,1	2,96
231	60	3,5	3,34
233	60	4	3,81
231	60	3,8	3,62
232	60	4	3,81
292	75	5	3,81
293	75	5,6	4,27
292	75	5,3	4,04
293	75	5,2	3,97
293	75	5,5	4,19
292	75	5,6	4,27

La elaboración de las pruebas para la comparación de modelos de control de movimiento en la morfología tipo ELE se creó un entorno como se ve en la Figura 48 y Figura 49, allí se ven tres

medidas con respecto al punto inicial del robot, estas medidas hacen referencia a las distintas distancias de referencia que se tomaron para realizar las pruebas de funcionamiento. La primera delimitación de derecha a izquierda es la línea de los 50cm, la segunda hace referencia a la línea de los 75cm y por último se encuentra la de los 100cm. Adicionalmente, en la Figura 38 se observa el cubo que es la representación del obstáculo, este fue ubicado a una distancia de 50cm, 75cm y 100cm.

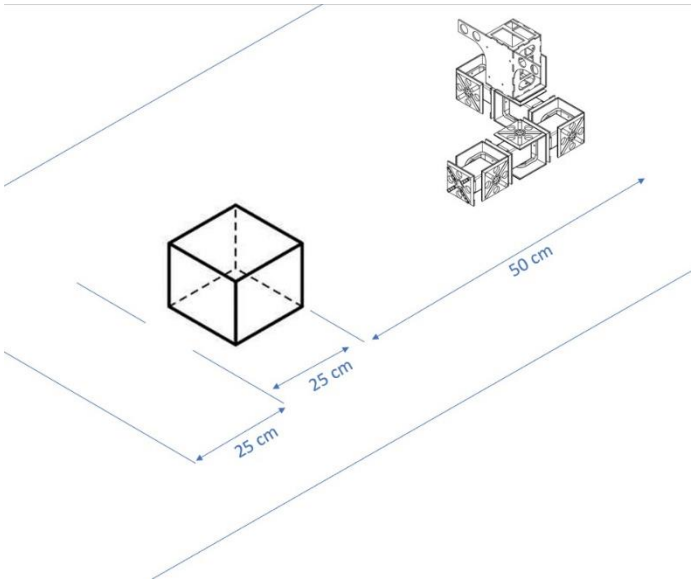


Figura 48. Plano de visualización del entorno de pruebas para la morfología tipo ELE con evasión de obstáculo. Elaboración propia.

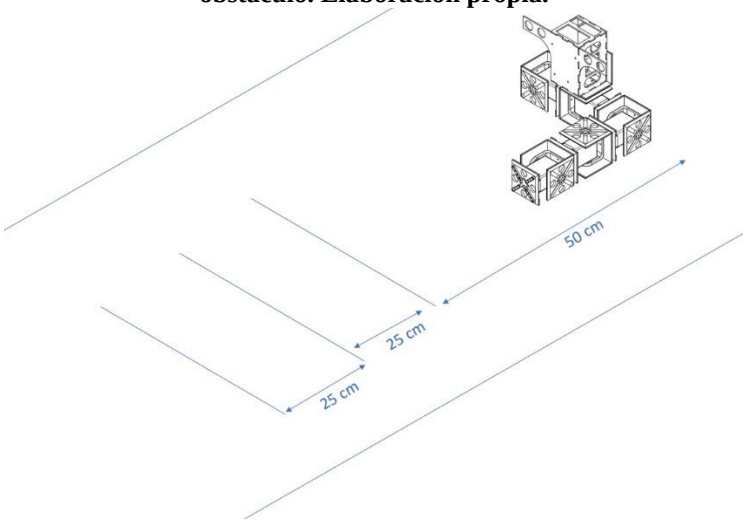


Figura 49. Plano de visualización del entorno de pruebas para la morfología tipo ELE en entorno llano. Elaboración propia.

En la Tabla 42 y Tabla 43 se resumen los resultados de las cinco pruebas realizadas para la morfología tipo ELE para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 50cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro

obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la derecha del robot.

Se observa que el tiempo de desplazamiento en promedio con el control de movimiento por tablas de movimiento es de 62.2 segundos, mientras que el promedio por control RNA es de 55.2 segundos, esto es 7 segundos menos que lo obtenido por tablas de movimiento como se observa en la Figura 50.

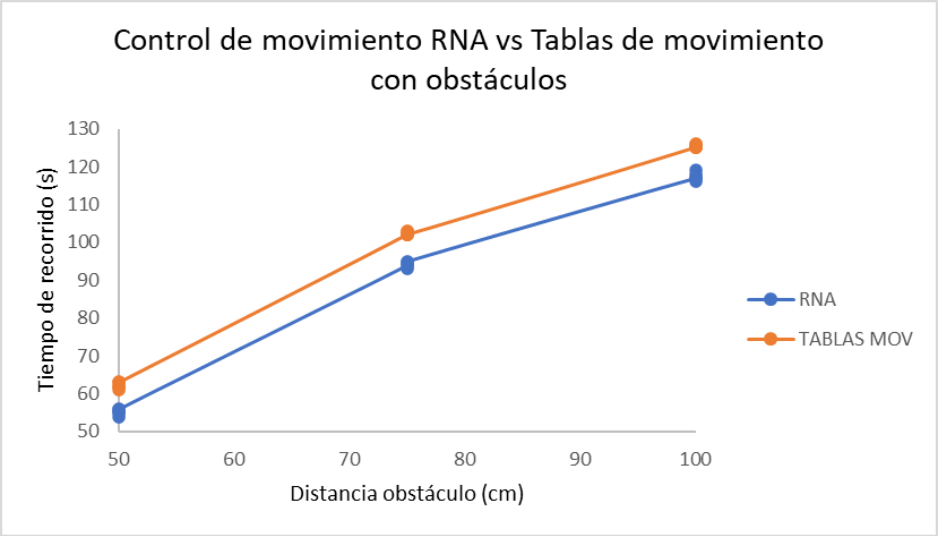


Figura 50. Representación gráfica morfología tipo ELE experimento con obstáculos. Elaboración propia.

Tabla 42. Resultados experimento morfología tipo ELE con obstáculo a 50cm en la derecha, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología tipo ELE (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
55	50	SI	14	Izquierda	Izquierda
54	50	SI	14	Izquierda	Izquierda
56	50	SI	15	Izquierda	Izquierda
55	50	SI	14	Izquierda	Izquierda
56	50	SI	14	Izquierda	Izquierda

Tabla 43. Resultados experimento morfología tipo ELE con obstáculo a 50cm en la derecha, con control de movimiento por tabla de movimiento. Elaboración propia.

Detección de obstáculos morfología tipo ELE (tabla de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
62	50	SI	15	Izquierda	Izquierda
61	50	SI	16	Izquierda	Izquierda

62	50	SI	16	Izquierda	Izquierda
63	50	SI	17	Izquierda	Izquierda
63	50	SI	15	Izquierda	Izquierda

En la Tabla 44 y Tabla 45 se resumen los resultados de las cinco pruebas realizadas para la morfología tipo ELE para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 75cm del robot y se tomaron como medidas de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la derecha del robot.

Se observa que en promedio el tiempo de desplazamiento a la distancia de evasión de obstáculo con el control de movimiento por tablas de movimiento es de 102.4 segundos, mientras que el promedio por control RNA es de 94.2 segundos como se observa en la Figura 50, esto es 8.2 segundos menos que lo obtenido por tablas de movimiento.

Tabla 44. Resultados experimento morfología tipo ELE con obstáculo a 75cm en la derecha, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología tipo ELE (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
94	75	SI	14	Izquierda	Izquierda
95	75	SI	15	Izquierda	Izquierda
93	75	SI	15	Izquierda	Izquierda
94	75	SI	14	Izquierda	Izquierda
95	75	SI	14	Izquierda	Izquierda

Tabla 45. Resultados experimento morfología tipo ELE con obstáculo a 75cm en la derecha, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos morfología tipo ELE (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
102	75	SI	17	Izquierda	Izquierda
103	75	SI	16	Izquierda	Izquierda
102	75	SI	17	Izquierda	Izquierda
103	75	SI	15	Izquierda	Izquierda
102	75	SI	17	Izquierda	Izquierda

La Tabla 46 y Tabla 47 resumen los resultados de las cinco pruebas realizadas para la morfología tipo ELE para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente.

En estas pruebas se ubicó un obstáculo a 100cm del robot y se tomaron como medidas de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la derecha del robot.

Se observa que el tiempo de desplazamiento promedio con el control de movimiento por tablas de movimiento es de 125.4 segundos, mientras que el promedio por control RNA es de 117.4 segundos como se ve en la Figura 50, esto es 8 segundos menos que lo obtenido por tablas de movimiento.

Tabla 46. Resultados experimento morfología tipo ELE con obstáculo a 100cm en la derecha, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología tipo ELE (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
117	100	SI	15	Izquierda	Izquierda
116	100	SI	15	Izquierda	Izquierda
118	100	SI	16	Izquierda	Izquierda
117	100	SI	16	Izquierda	Izquierda
119	100	SI	16	Izquierda	Izquierda

Tabla 47. Resultados experimento morfología tipo ELE con obstáculo a 100cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos morfología tipo ELE (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
125	100	SI	16	Izquierda	Izquierda
126	100	SI	17	Izquierda	Izquierda
126	100	SI	15	Izquierda	Izquierda
125	100	SI	17	Izquierda	Izquierda
125	100	SI	16	Izquierda	Izquierda

En la Tabla 48 y Tabla 49 se resumen los resultados de las cinco pruebas realizadas para la morfología tipo ELE para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas se ubicó un obstáculo a 50cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la izquierda del robot.

El tiempo de desplazamiento promedio con el control de movimiento por tablas de movimiento es de 62.2 segundos, en cambio el promedio por control RNA es de 54.2 segundos, esto es 8 segundos menos que lo obtenido por tablas de movimiento.

Tabla 48. Resultados experimento morfología tipo ELE con obstáculo a 50cm en la izquierda, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología tipo ELE (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
54	50	SI	15	Derecha	Derecha
55	50	SI	14	Derecha	Derecha
53	50	SI	15	Derecha	Derecha
55	50	SI	16	Derecha	Derecha
54	50	SI	14	Derecha	Derecha

Tabla 49. Resultados experimento morfología tipo ELE con obstáculo a 50cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos morfología tipo ELE (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
61	50	SI	16	Derecha	Derecha
63	50	SI	15	Derecha	Derecha
61	50	SI	15	Derecha	Derecha
62	50	SI	15	Derecha	Derecha
64	50	SI	16	Derecha	Derecha

En la Tabla 50 y Tabla 51 se resumen los resultados de las cinco pruebas realizadas para la morfología cadena para los algoritmos basados propuestos en el presente documento. En estas pruebas se ubicó un obstáculo a 75cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la izquierda del robot.

Se observa que el tiempo de desplazamiento hasta evadir el obstáculo, en promedio con el control de movimiento por tablas de movimiento es de 103 segundos, mientras que el promedio por control RNA es de 86.2 segundos, esto es 16.8 segundos menos que lo obtenido por tablas de movimiento.

Tabla 50. Resultados experimento morfología tipo ELE con obstáculo a 75cm en la izquierda, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología tipo ELE (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
53	75	SI	15	Derecha	Derecha
94	75	SI	14	Derecha	Derecha

95	75	SI	15	Derecha	Derecha
95	75	SI	16	Derecha	Derecha
94	75	SI	15	Derecha	Derecha

Tabla 51. Resultados experimento morfología tipo ELE con obstáculo a 75cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos morfología tipo ELE (tabla de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
104	75	SI	15	Derecha	Derecha
103	75	SI	17	Derecha	Derecha
104	75	SI	16	Derecha	Derecha
101	75	SI	15	Derecha	Derecha
103	75	SI	22	Derecha	Derecha

En la Tabla 52 y Tabla 53 se resumen los resultados de las cinco pruebas realizadas para la morfología tipo ELE para los algoritmos basados en tablas de movimiento y redes neuronales, respectivamente. En estas pruebas el obstáculo se ubicó a 100cm del robot y se tomaron como mediciones de comparación entre los modelos el tiempo de desplazamiento, si se detectó el obstáculo, distancia a la que se evadió el obstáculo, sentido de giro esperado y sentido de giro obtenido, estos últimos de acuerdo a la posición del obstáculo, que en este caso estaba ubicado a la izquierda del robot.

El tiempo de desplazamiento hasta la evasión del obstáculo, en promedio con el control de movimiento por tablas de movimiento es de 127 segundos, mientras que el promedio por control RNA es de 117.2 segundos, esto es 9.8 segundos menos que lo obtenido por tablas de movimiento.

Tabla 52. Resultados experimento morfología tipo ELE con obstáculo a 100cm en la izquierda, con control de movimiento por RNA. Elaboración propia.

Detección de obstáculos en morfología tipo ELE (RNA)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
116	100	SI	15	Derecha	Derecha
118	100	SI	15	Derecha	Derecha
117	100	SI	14	Derecha	Derecha
118	100	SI	16	Derecha	Derecha
117	100	SI	15	Derecha	Derecha

Tabla 53. Resultados experimento morfología tipo ELE con obstáculo a 100cm en la izquierda, con control de movimiento por tablas de movimiento. Elaboración propia.

Detección de obstáculos morfología tipo ELE (tablas de movimiento)					
Tiempo de desplazamiento (s)	**Distancia obstáculo (cm)**	**Detección obstáculo**	**Distancia de evasión con respecto al obstáculo (cm)**	**Sentido de giro esperado**	**Sentido de giro obtenido**
127	100	SI	15	Derecha	Derecha
128	100	SI	23	Derecha	Derecha
127	100	SI	17	Derecha	Derecha
126	100	SI	16	Derecha	Derecha
127	100	SI	16	Derecha	Derecha

En la Tabla 54 y Tabla 55 se resumen los resultados de las cinco pruebas realizadas para la morfología tipo ELE para los algoritmos propuestos. En estas pruebas se hicieron tres delimitaciones, la primera a 50cm, la segunda a 75cm y la última a 100cm del robot y se tomaron las mismas mediciones que con la morfología cadena.

El tiempo de desplazamiento hasta la evasión del obstáculo, en promedio con el control de movimiento por tablas de movimiento es de 67.8 segundos para una distancia de 50cm, 111.5 segundos para una distancia de 75cm y 134.5 segundos para una distancia de 100cm, mientras que el promedio por control RNA es de 61 segundos para una distancia de 50cm, 97.5 segundos para una distancia de 75cm y 124 para una distancia de 100cm, con esto se obtiene que para la morfología tipo ELE el rendimiento de la RNA es mejor al de las tablas de movimiento, ya que en las tres distancias el tiempo de desplazamiento promedio fue menor como se ve en la Figura 51.

También, se evidencia que bajo el control de movimiento por tablas de movimiento el promedio del ángulo de desfase para 50cm fue de 8.86°, para 75cm fue de 18.66° y para 100cm fue de 24.78°; en contra parte, con el ángulo de desfase con el control de movimiento de la RNA fue de 7.29° para una distancia de 50cm, 10.94° para una distancia de 75cm y de 16.59° para una distancia de 100cm como se ve en la Figura 52. Con lo cual se observa que con la RNA se disminuyó el promedio de desfase del robot permitiendo que el desplazamiento sea más recto.

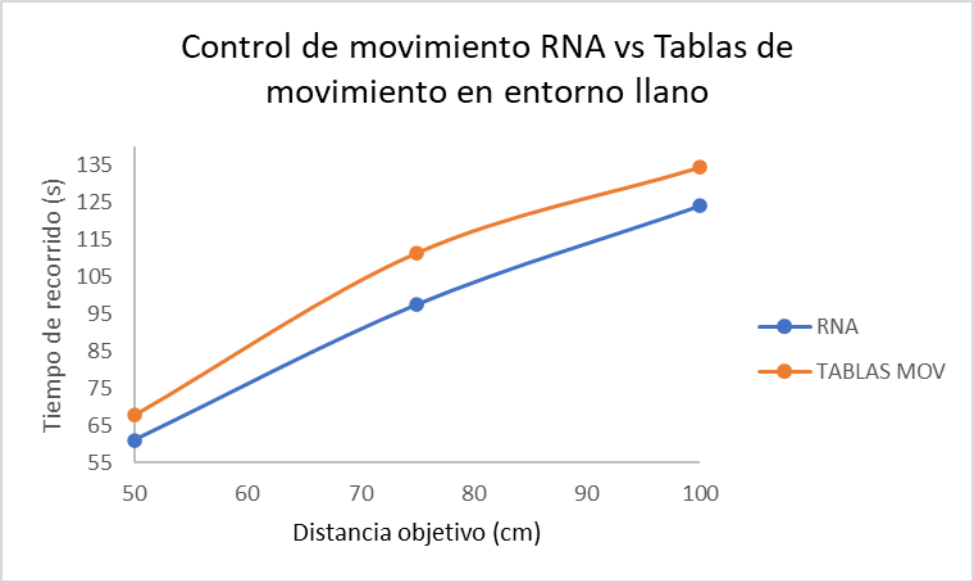


Figura 51. Representación gráfica morfología tipo ELE experimento en entorno llano. Elaboración propia.

Tabla 54. Resultados experimento morfología tipo ELE en entorno llano, con control de movimiento por RNA. Elaboración propia.

Morfología tipo ELE recorrido sin obstáculos (RNA)			
Tiempo (s)	**Distancia H (cm)**	**Distancia Y (cm)**	**Angulo de desfase °**
60	50	6	6,84
58	50	7	7,97
62	50	5	5,71
62	50	6	6,84
63	50	8	9,09
100	75	13	9,83
98	75	15	11,31
94	75	14	10,57
95	75	13	9,83
98	75	14	10,57
100	75	18	13,50
125	100	25	14,04
124	100	28	15,64
126	100	27	15,11
124	100	30	16,70
123	100	38	20,81
122	100	31	17,22

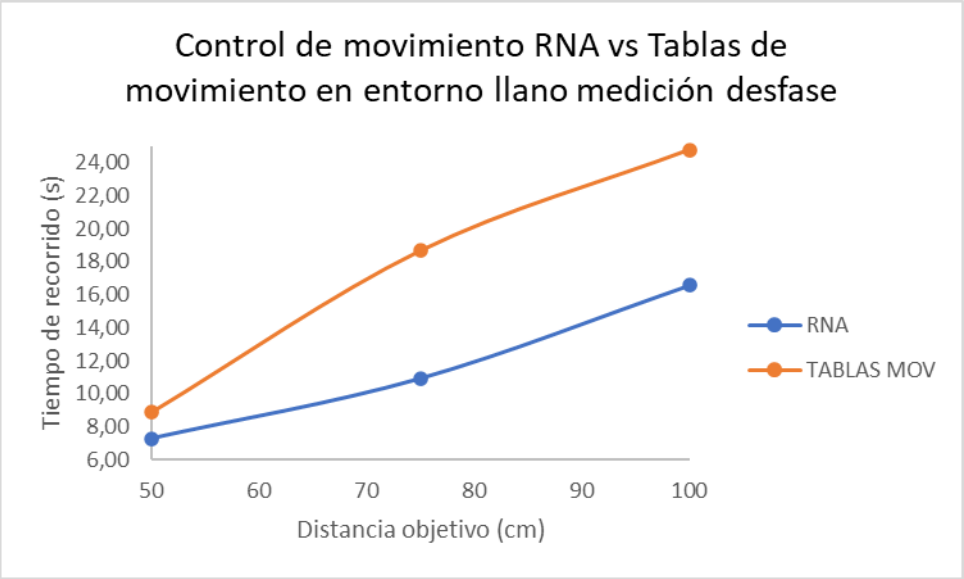


Figura 52. Representación gráfica morfología tipo ELE experimento en entorno llano medición ángulo de desfase. Elaboración propia.

Tabla 55. Resultados experimento morfología tipo ELE en entorno llano, con control de movimiento por tablas de movimiento. Elaboración propia.

Morfología tipo ELE recorrido sin obstáculos (tablas de movimiento)			
Tiempo (s)	**Distancia H (cm)**	**Distancia Y (cm)**	**Angulo de desfase °**
68	50	7	7,97
67	50	8	9,09
67	50	7	7,97
68	50	8	9,09
69	50	9	10,20
110	75	23	17,05
112	75	25	18,43
111	75	24	17,74
110	75	27	19,80
112	75	26	19,12
114	75	27	19,80
130	100	44	23,75
132	100	47	25,17
136	100	48	25,64
134	100	46	24,70
138	100	47	25,17
137	100	45	24,23

8. Conclusiones y Recomendaciones

Se debe mejorar el sistema de interconexión de los módulos, en algunas superficies tienden a desprenderse los módulos, también sucede al realizar algunos patrones de movimiento.

Al implementar una velocidad mayor en los motores y un tiempo de espera en el envío de la trama, esto no permite que el robot tenga un movimiento natural y que el robot se desplace distancias más cortas. Así como, causa un sobrecalentamiento en los actuadores lo que provoca un menor tiempo de ejecución ya que estos tienen un sistema de protección contra temperaturas altas y dejan de ejecutar los comandos que se le envían.

Se implementó una RNA por cada posible entrada y por cada movimiento debido a la complejidad de entrenar la red con cuatro posibles entradas y todos los movimientos de una sola morfología, lo que provocaba que la red no se entrenara adecuadamente y obteniendo una precisión baja.

En los experimentos que se realizaron se pudo observar que en las morfologías de cadena y tipo ELE se puede realizar un barrido para detectar obstáculos a un ángulo mayor a la posición 0 de los sensores, en cambio en la morfología de bloque los sensores solo pueden estar en la posición 0 como se ve en las imágenes de la sección 7.1, donde el controlador local se encuentra en un módulo que está posicionado para rotarse según la morfología.

Los parámetros de entrenamiento de este tipo de aplicaciones, como el control de movimiento de robots modulares ápodos no tiene un estudio destallado y solo hasta encontrar la configuración correcta por medio de prueba y error se puede llegar a obtener un entrenamiento adecuado como en este caso, donde la función de perdida que permitió el entrenamiento del modelo fue la Poisson, así como la función de activación de la capa de entrada y salida fue la tipo relu, y el optimizador con mejores resultados y que permitió la precisión fue el nadam; para obtener la precisión deseada también fue necesario emplear múltiples capas ocultas con distintas funciones de activación y diversas cantidades de neuronas por capa.

La implementación de redes neuronales multicapa de aprendizaje profundo para el control de movimiento del robot EMERGE en las morfologías cadena, bloque y tipo ELE, permite la disminución de tiempos de desplazamiento en entornos con obstáculos, así como un comportamiento recto del desplazamiento. Por otro lado, para las morfologías cadena, tipo ELE y bloque los parámetros de salida fueron favorecedores también en este tipo de entorno para el tiempo de desplazamiento al punto objetivo y el desfase del punto de origen en el eje Y, como se observa en la sección 7.4.

Se debe tener en cuenta el rango de lectura de los sensores para detectar un obstáculo, así como, la base donde están ubicados estos no rotan y el campo de lectura de los sensores utilizados funciona únicamente en ángulo recto lo que delimita la detección de obstáculos. En la morfología tipo ELE la detección de obstáculos presenta intermitencias dado la ubicación de los sensores. Se sugiere la implementación de un sistema de detección de obstáculos más robusto para mejorar la precisión de esta funcionalidad en futuras investigaciones.

La morfología que en entorno llano tuvo el menor tiempo de desplazamiento hasta el límite objetivo con el algoritmo de las redes neuronales fue el tipo cadena y un desface intermedio con respecto a las otras morfologías; por otro lado, el tipo bloque tuvo un menor desface con respecto al punto inicial y el punto final en el eje Y, pero el mayor tiempo de desplazamiento. De la misma manera, el comportamiento de la configuración cadena tuvo el mejor desempeño en entornos con obstáculos, dada la facilidad de la morfología para cambiar de dirección en pocos pasos con respecto a las demás.

9. Referencias

Adamu, E. I., Afolayan, M. O., Umaru, S., & Garba, D. K. (2018). The modelling and control of the drive system of an Ackermann Robot using GA optimization. *Nigerian Journal of Technology (NIJOTECH)*, *37*(4), 1008–1014. https://doi.org/10.4314/njt.v37i4.21

Aguilar, D., & Vivas, A. (2009). *Modelado y control de un robot bípedo de nueve grados de libertad*. *8*. http://www.unicauca.edu.co/ai/publicaciones/Aguilar_ACA09.pdf

Ahmadzadeh, H., Masehian, E., Asadpour, M., Ahmadzadeh, H., & Asadpour, M. (2016). Modular Robotic Systems: Characteristics and Applications. *J Intell Robot Syst*, *81*, 317–357. https://doi.org/10.1007/s10846-015-0237-8

Alattas, R. J., Patel, S., & Sobh, T. M. (2019). Evolutionary Modular Robotics: Survey and Analysis. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *95*(3–4), 815–828. https://doi.org/10.1007/s10846-018-0902-9

Aliexpress. (2022). *ROS robot Ackerman diferencial coche no tripulado McNum rueda Moveit brazo robótico Raspberry*. https://es.aliexpress.com/item/1005002541064402.html?af=6659874&aff_fcid=48a5e92f92d14620a55ed6958c4cbb09-1644288239522-02560-_uQvdYY&aff_fsk=_uQvdYY&aff_platform=api-new-link-generate&sk=_uQvdYY&aff_trace_key=48a5e92f92d14620a55ed6958c4cbb09-164428823952

Alpaydin, E. (2014). *Introduction to Machine Learning* (T. Dietterich, C. Bishop, D. Heckerman, M. Jordan, & M. Kearns (eds.); Third). The MIT Press.

Amazon Web Services (AWS). (n.d.). *¿Qué es la inteligencia artificial (IA)?: Amazon Web Services*. Retrieved March 6, 2023, from https://aws.amazon.com/es/machine-learning/what-is-ai/

Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, *9*(3), 31–37. https://doi.org/10.1109/37.24809

Arenas Morantes, A. A., Rincón Cevallos, A. P., Pedraza Martínez, L. F., & Hernández Martínez, H. A. (2020). Desarrollo de una aplicación para simular los movimientos del robot modular EMERGE en configuración tipo cadena. *Revista Vínculos*, *17*(2), 112–119. https://doi.org/10.14483/2322939x.17352

Aurelian, O. (2015). The geometric and kinematic model of RTTRR smal-sized modular robot. *Acta Technica Napocensis*, *58*(Iv), 513–518.

Baca, J., Hossain, S. G. M., Dasgupta, P., Nelson, C. A., & Dutta, A. (2014). ModRED: Hardware design and reconfiguration planning for a high dexterity modular self-reconfigurable robot for extra-terrestrial exploration. *Robotics and Autonomous Systems*, *62*(7), 1002–1015. https://doi.org/10.1016/j.robot.2013.08.008

Behal, A., Dixon, W., Dawson, D. M., & Xian, B. (2009). Machine Learning Algorithms in Bipedal Robot Control. In *Lyapunov-Based Control of Robotic Systems* (Vol. 42, Issue 5, pp. 9–36). CRC Press. https://doi.org/10.1201/9781420006278-2

Belke, C. H., & Paik, J. (2017). Mori: A Modular Origami Robot. *IEEE/ASME Transactions on Mechatronics*, *22*(5), 2153–2164. https://doi.org/10.1109/TMECH.2017.2697310

Bhattacharya, S., Dutta, S., Maiti, T. K., Miura-Mattausch, M., Navarro, D., & Mattausch, H. J. (2018). Machine learning algorithm for autonomous control of walking robot. *2018*

International Symposium on Devices, Circuits and Systems, ISDCS 2018, 1–4. https://doi.org/10.1109/ISDCS.2018.8379644

Brunete, A, Gambao, E., Torres, J. E., & Hernando, M. (2006). A 2 DoF servomotor-based module for pipe inspection modular micro-robots. *IEEE International Conference on Intelligent Robots and Systems*, 1329–1334. https://doi.org/10.1109/IROS.2006.281898

Brunete, Alberto, Ranganath, A., Segovia, S., Perez De Frutos, J., Hernando, M., & Gambao, E. (2017). Current trends in reconfigurable modular robots design. *International Journal of Advanced Robotic Systems*, 1–21. https://doi.org/10.1177/1729881417710457

Calle, E., Ávila, I., & Zambrano, J. (2007). Diseño e Implementación de un Robot Móvil Cuadrúpedo. *Revista Tecnológica ESPOL*, *20*(1), 65–72.

Chen, I.-M., & Yim, M. (2008). Modular Robots. In B. Siciliano & O. Khatib (Eds.), *Springer Handbook of Robotics* (1st ed., pp. 531–542). Springer, Berlin, Heidelberg. https://doi.org/https://doi.org/10.1007/978-3-540-30301-5

Conde González, N. F., & Mahecha Mora, L. F. (2021). *Esquema de control de movimiento basado en redes neuronales y procesamiento de imágenes para robots modulares tipo cadena*. Universidad Distrital Francisco José de Caldas.

ELECTROSTORE. (2019). *Llanta omnidireccional mecanum derecha 97mm*. https://grupoelectrostore.com/shop/estructuras-y-partes-no-electronicas-de-robots/llantas-estructuras-y-partes-no-electronicas-de-robots/llanta-omnidireccional-mecanum-derecha-97mm/

Ellips Masehian, H. A. (2015). Modular robotic systems: Methods and algorithms for abstraction, planning, control, and synchronization. *Artificial Inteligence*, 27–64. https://doi.org/http://dx.doi.org/10.1016/j.artint.2015.02.004

Es.dhgate.com. (2022). *Compra Chasis Plataforma Móvil Para Omni Direccional De Tres Ruedas*. https://es.dhgate.com/product/omni-directional-three-wheel-robot-car-chassis/553278268.html

FBI. (n.d.). *Science and Technology Branch — FBI*. Retrieved February 7, 2022, from https://www.fbi.gov/about/leadership-and-structure/science-and-technology-branch

Felipe, L., Giraldo, L. F., & Delgado, E. (2006). Cinemática Inversa de un Brazo Robot Utilizando Algoritmos Genéticos. *Avances En Sistemas e Informática*, *3*(1), 29–34.

Filio García, M., González Zárate, R., & García Lozano, H. (2019). Diseño y construcción de un robot móvil síncrono. *ConCiencia*, *6*, 7–10.

Gómez, H., Porras, N., & Pedraza, L. (2020). *Sistema de comunicación basado en radio definido por software para un robot modular*. Universidad Distrital Francisco José de Caldas.

González Gomez, J. (2008). *Robótica modular y locomoción: aplicación a robots ápodos*. Universidad Autónoma de Madrid.

Gonzalez Gomez, J., Boemo, E., Zhang, H., & Zhang, J. (2006). *Locomotion Capabilities of a Modular Robot with Eight Pitch-Yaw-Connecting Modules*. http://www.iearobotics.com/personal/juan/publicaciones/art14/html/node2.html

Hernández Martinez, H. A. (2018). *Estimación de una serie de movimientos utilizando un algoritmo de optimización bio-inspirado para la operación de manera autónoma y On-Line de una plataforma Multi-Robot (Caso robot modular)*. Universidad Nacional de Colombia.

Hernández Martínez, H., Montiel Ariza, H., & Gómez Castillo, H. (2020). Evaluation of the Performance of a Deep Learning Neural Network when Controlling a Modular Chain- Type Robot. *International Journal of Mechanical & Mechatronics Engineering*, *20*(05), 84–90.

Hrbáček, J., Ripel, T., & Krejsa, J. (2010). Ackermann mobile robot chassis with independent rear wheel drives. In IEEE (Ed.), *Proceedings of EPE-PEMC 2010 - 14th International Power Electronics and Motion Control Conference* (pp. 46–51). IEEE. https://doi.org/10.1109/EPEPEMC.2010.5606853

Ibarra-Peñaranda, M. L., Duque-Suárez, O. M., & Duque-Suarez, M. C. (2021). Sistema de control cinemático guiado y colaborativo por percepción de las trayectorias de las extremidades superiores. *Aibi Revista de Investigación, Administración e Ingeniería*, *8*(S1), 124–151. https://doi.org/10.15649/2346030x.2394

Ibarra Peñaranda, M. L., Duque Suárez, O. M., & Duque Suárez, M. C. (2021). Control Cinemático De Un Robot Guiado Por Las Trayectorias De Las Extremidades Superiores Del Humano. *South Florida Journal of Development*, *2*(1), 871–889. https://doi.org/10.46932/sfjdv2n1-064

Iporre Rivas, A. (2014). Modelo matemático de un robot movil diferencial. *Research Gate*.

ISO/IEC. (n.d.). *ISO - ISO/IEC AWI TS 8200 - Information technology — Artificial intelligence — Controllability of automated artificial intelligence systems*. Retrieved February 11, 2022, from https://www.iso.org/standard/83012.html?browse=tc

ISO - ISO/IEC TR 24029-1:2021 - Artificial Intelligence (AI) — Assessment of the robustness of neural networks — Part 1: Overview, Pub. L. No. ISO/IEC TR 24029-1:2021 (2021). https://www.iso.org/standard/77609.html?browse=tc

Jiménez Cámara, A. (2012). *Control centralizado de flotas de robots*. Carlos III de Madrid.

Keras. (n.d.). *Keras: Deep Learning for humans*. Retrieved April 1, 2023, from https://keras.io/

Kuwayama, K., Kato, S., Kunitachi, T., & Itoh, H. (n.d.). *Motion control for humanoid robots based on the motion phase decision tree learning*. https://doi.org/10.1109/MHS.2004.1421294

Lee, W. H., & Sanderson, A. C. (2002). Dynamic Rolling Locomotion and Control of Modular Robots. *IEEE Transactions on Robotics and Automation*, *18*(1), 32–41. https://doi.org/10.1007/978-3-319-32552-1_22

López García, S. R. (2011). *Calibración de robots de locomoción diferencial por el método de mínimos cuadrados empleando la técnica de suavizado y cartografía incremental*. Universidad Michoacana de San Nicolás de Hidalgo.

Magdalena Layos, L. (1994). *Estudio de la coordinación inteligente en robots bípedos: aplicación de lógica borrosa y algoritmos genéticos* [Universidad Politécnica de Madrid]. https://dialnet.unirioja.es/servlet/tesis?codigo=9581&info=resumen&idioma=SPA

Maglogiannis, I., Macinture, J., Iliadis, L., & All, E. (2021). Artificial intelligence applications and innovations. In I. Maglogiannis, J. Macinture, & L. Iliadis (Eds.), *IFIP Advances in Information and Communication Technology* (Vol. 154). Springer Nature Switzerland AG. https://doi.org/ISBN 978-3-https://doi.org/10.1007/978-3-030-79150-6

Mannion, P., Duggan, J., & Howley, E. (2016). Autonomic Road Transport Support Systems. *Autonomic Road Transport Support Systems*, 47–66. https://doi.org/10.1007/978-3-319-25808-9

Marbach, D., & Ijspeert, A. J. (2004). *Co-evolution of Configuration and Control for Homogenous Modular Robots*.

Marín, L., Soriano, Á., Mayans, V., Vallés, M., Valera, Á., & Albertos, P. (n.d.). *Localización asistida por GPS para robots móviles en configuración Ackermann de recursos limitados*.

Martín del Brío, B., & Sanz Molina, A. (2001). Fundamentos de las redes neuronales artificiales. In Alfaomega (Ed.), *Redes Neuronales y Sistemas Difusos* (2da ed., pp. 3–37). Alfaomega.

Martínez, H. H., Ariza, H. M., & Castillo, H. G. (2020). Evaluation of the Performance of a Deep Learning Neural Network when Controlling a Modular Chain- Type Robot. *International Journal of Mechanical & Mechatronics Engineering*, *20*(05), 84–90.

Martinez Heras, J. (2020, October 10). *15 Librerías de Python para Machine Learning*. https://www.iartificial.net/librerias-de-python-para-machine-learning/#Librerias_de_Python_para_Deep_Learning

Martínez, S., & Sisto, R. (2009). *Control y Comportamiento de Robots Omnidireccionales* [Universidad de la República Montevideo]. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992018000100014

Matich, D. J. (2001). Redes Neuronales: Conceptos Básicos y Aplicaciones. In *Historia* (p. 55). ftp://decsai.ugr.es/pub/usuarios/castro/Material-Redes-Neuronales/Libros/matich-redesneuronales.pdf

Méndez Polanco, J. A., Amaro Camargo, E., Muñoz Meléndez, A., & Morales Manzanarez, E. F. (2006). *Diseño y Construccion de un robot Ápodo*. *February*.

Miño Rubiano, W. A. (2017). *Simulación e implementación de sistema robotico en arquitectura tipo rueda utilizando robotica modular* [Universidad Militar Nueva Granada]. https://repository.unimilitar.edu.co/handle/10654/16081

Morantes, A. A. A., Cevallos, A. P. R., Pedraza Martínez, L. F., & Hernández, H. A. (2020). *Desarrollo de una aplicacion para simular los movientos del robot modular EMERGE en configuracion tipo cadena*. 88–92.

Moreno García, R. (2010). *Diseño y Simulación de un Algoritmo para el Control de un Robot Modular tipo Cadena*. http://www.bdigital.unal.edu.co/5111/

Moreno, R., Liu, C., Faina, A., Hernandez, H., & Gomez, J. (2017). The EMeRGE modular robot, an open platform for quick testing of evolved robot morphologies. *The Genetic and Evolutionary Computation Conference Companion*, *July*, 71–72. https://doi.org/10.1145/3067695.3075616

Mortes Esquer, J. (2021). *Construcción , simulación y programación de un robot cuadrúpedo multipropósito de código abierto*. Universidad de Alicante.

Nascimento Martins, F., Santos Brandao, A., Carelli, R., Sarcinelli Filho, M., & Bastos Filho, T. F. (2008). *Control Descentralizado de una Formación Líder-Seguidor con Compensación de la Dinámica*. 1–6.

Ojeda, I. B., Tolu, S., Pacheco, M., Pacheco, M., Christensen, D. J., & Lund, H. H. (2017). A Combination of Machine Learning and Cerebellar-like Neural Networks for the Motor Control and Motor Learning of the Fable Modular Robot. *Journal of Robotics, Networking and Artificial Life*, *4*(1), 62–66. https://doi.org/10.2991/jrnal.2017.4.1.14

Oracle Colombia. (n.d.). *¿Qué es la inteligencia artificial (IA)?* . Retrieved March 6, 2023, from https://www.oracle.com/co/artificial-intelligence/what-is-ai/

Pérez Machorro, J. (2009). *Generación de locomoción de un robot hexápodo usando dos células neuronales analógicas*. Centro Nacional de Investigación y Desarrollo Tecnológico.

Pouya, S., Aydin, E., Möckel, R., & Ijspeert, A. J. (2011). Locomotion Gait Optimization For Modular Robots; Coevolving Morphology and Control. *Procedia Computer Science*, *7*, 320–322. https://doi.org/10.1016/J.PROCS.2011.09.084

Reddy, C. H. S. S., Patlolla, S., Agrawal, A., & Anupama, K. R. (2017). HexaMob-A hybrid modular robotic design for implementing biomimetic structures. *Robotics*, *6*(4). https://doi.org/10.3390/robotics6040027

Reyes García, C. A., Morales Manzanares, E. F., Escalante Balderas, H. J., & Torres García, A. A. (2018). Aprendizaje e Inteligencia Computacional. In *Academia Mexicana de Computación* (Primera). Academia Mexicana de Computación. http://amexcomp.mx/files/AprendizajeInteligenciaComputacional.pdf

Rico, E. M., Hernández Riveros, J. A., Cardona, J., & Correa, S. (2013). Implementación de CPG en robots de locomoción continua y discreta. *Vinculos*, *10*(1), 74–85. https://doi.org/10.14483/2322939X.4699

Robles Atuesta, J. M. (2020). Diseño y prototipado del mecanismo de locomoción para un robot cuadrúpedo [Universidad de los Andes]. In *Universidad de los Andes*. http://hdl.handle.net/1992/45035

ROBOTICS. (n.d.). *Dynamixel AX-12A* (AX-12+, AX-12A). Retrieved May 1, 2023, from https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

Rodríguez-Calderón, J. P., Ramos-Parra, M. F., & Peña-Giraldo, M. V. (2015). Simulación de un Robot Hexápodo Bioinspirado en el Tenebrio. *Lámpsakos*, *14*, 33–39. https://doi.org/10.21501/21454086.1636

Rodríguez Lombana, G. A. (2019). DISEÑO DE ALGORITMO DE AUTOCONFIGURACIÓN CONSIDERANDO DOS ARQUITECTURAS (1D Y 2D) PARA SISTEMA ROBÓTICO MODULAR. In *Universidad Militar Nueva Granada* (Vol. 8, Issue 5). Universidad Militar Nueva Granada.

Segway. (n.d.). *SEGWAY*. Retrieved February 11, 2022, from https://segwayninebot.co/

Servotronik. (2022). *Robot hexapodo metalico*. https://www.servotronik.com.co/index.php/producto/robot-hexapodo-metalico/

Shigemune, H., Maeda, S., Hara, Y., Hosoya, N., & Hashimoto, S. (2016). Origami Robot: A Self-Folding Paper Robot With an Electrothermal Actuator Created by Printing. *IEEE/ASME Transactions on Mechatronics*, *21*(6), 2746–2754. https://doi.org/10.1109/TMECH.2016.2593912

TensorFlow - Google. (n.d.). *Module: tf.keras | TensorFlow v2.12.0*. Retrieved April 1, 2023, from https://www.tensorflow.org/api_docs/python/tf/keras

Toapanta Paucar, C. X. (2019). *Implementación de un prototipo de robot bípedo con tecnología XBEE para la detección de obstáculos predeterminados*. Universidad Tecnológica Israel.

Trappey, A. J. C., Trappey, C. V., Wu, J. L., & Wang, J. W. C. (2020). Intelligent compilation of patent summaries using machine learning and natural language processing techniques. *Advanced Engineering Informatics*, *43*(June 2019), 1–13. https://doi.org/10.1016/j.aei.2019.101027

Vele G, O. L. (2005). *Cinemática inversa de un robot bípedo*. *November*, 1–20.

Xataka.com. (2020). *El fin de una era: Segway deja de fabricar su mítico vehículo dos décadas después de su invención*. https://www.xataka.com/vehiculos/fin-era-segway-deja-fabricar-su-mitico-vehiculo-dos-decadas-despues-su-invencion

Ye, C., Chen, N., Chen, L., & Jiang, C. (2018). A Variable-Scale Modular 3D Printing Robot of Building Interior Wall. *IEEE International Conference on Mechatronics and Automation (ICMA)*, 1818–1822. https://doi.org/10.1109/ICMA.2018.8484433

Yim, M. (1994). *Locomotion with a unit-modular reconfigurable robot* (Issue December) [Universidad de Standford]. ftp://db.stanford.edu/pub/cstr.old/reports/cs/tr/95/1536/CS-TR-95-1536.pdf

Zamorano Martín, D. (2011). *Control locomotor multidireccional de un robot modular mediante*

Circuitos Generadores Centrales de Patrones Bioinspirados. Universidad Autónoma de Madrid.

10.Anexos

10.1. Anexo 1

Código Arduino controlador local

```
#include <DynamixelSerial.h>
#include <SoftwareSerial.h>
#include <SoftwareSerial.h>
//#include <HCSR04.h>

SoftwareSerial mySerial(12, 13); //RX,TX
//HCSR04 hc1(2,6); //initialisation class HCSR04 (trig pin , echo pin)
//HCSR04 hc2(5,8);
char cadena[25];
int
datos[7];//velocidad[0],angulo1[1],angulo2[2],angulo3[3],angulo4[4],angulo5[5]...
float dts=0;
int estados=0;
int temp=0;
int flag=0;
int num=0;
float distancia=0;
float distancia2=0;
void setup() {
// put your setup code here, to run once:
Dynamixel.setSerial(&Serial); // &Serial - Arduino UNO/NANO/MICRO, &Serial1,
&Serial2, &Serial3 - Arduino Mega
Dynamixel.begin(115200,1);  // Inicialize the servo at 1 Mbps and Pin Control 2
delay(500);
mySerial.begin(4800);
}

int IDENT(char x) {
  switch (x) {
  case '0':
    num=0;
    break;
  case '1':
    num=1;
    break;
  case '2':
    num=2;
    break;
  case '3':
```

```
    num=3;
    break;
  case '4':
    num=4;
    break;
  case '5':
    num=5;
    break;
  case '6':
    num=6;
    break;
  case '7':
    num=7;
    break;
  case '8':
    num=8;
    break;
  case '9':
    num=9;
    break;
}
  return num;
}



void loop() {

  if (mySerial.available()) { //Si está disponible
    while (mySerial.available() > 0) {
      char c = mySerial.read();
      if(flag==1){
        cadena[temp]=c;
      temp++;
      }
      if (c=='Z'){
        flag=1;
      }
      if(c=='D'){
        temp=0;
        estados=15;
        break;
        }
```

```
      if(c=='#'){
        temp=0;
        estados=1;
        flag=0;
        //Serial.println(cadena);
        break;
        }
    }
  }
  //Guardamos la lectura en una variable char

  switch (estados) {
  case 0:
  //esperamos datos
    break;
  case 1:
for (int t = 0; t < 6; t++) {
        datos[t]=IDENT(cadena[t*4])*100+IDENT(cadena[t*4+1])*10+IDENT(cadena[t*4+
2]);
      }
    estados=2;
    break;

  case 2:
    //limpiamos el vector de los datos
      for (int t = 0; t < 25; t++) {
        cadena[t]=' ';
      }
      delay(500);
      estados=3;

  case 3:

      for (int t = 1; t < 6; t++) {
        if(datos[t]>=250 and datos[t]<=900){
        Dynamixel.moveSpeed(t,datos[t],datos[0]);// modulo,angulo,velocidad
        delay(10);
        }
      }
      estados=20;


  break;
```

```
  case 15:
    distancia=0.0;
    distancia2=0.0;
    /*for (int t = 0; t < 6; t++){
    dts= hc.dist(0);
    distancia=distancia+dts;
    delay(30);
    }
    distancia=distancia/6;

    for (int t = 0; t < 6; t++){
    dts= hc.dist(1);
    distancia2=distancia2+dts;
    delay(30);
    }
    distancia2=distancia2/6;
    */
    //distancia= hc1.dist();
    delay(30);
    //distancia2= hc2.dist();
    delay(30);
    //mySerial.println(String(int(distancia))+","+String(int(distancia2))+"#");
    distancia=0;
    distancia2=0;
    delay(30);
    estados = 0 ;
  break;
  }


}
```

10.2. Anexo 2

Interfaz gráfica para entrenamiento de tablas de control de movimiento y algoritmo de control basado en tablas de movimiento.

```
import tkinter
import numpy
import math
import time
from tkinter import *
```

```
from tkinter import Tk
from tkinter import messagebox
from tkinter import constants
from tkinter import filedialog
from tkinter import Canvas
import serial
arduino = serial.Serial('COM3', 9600, timeout=1)
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
arduino = serial.Serial('/dev/ttyS0', 9600, timeout=1)
global listaM1, listaM2, listaM3
listaM1=[]
listaM2=[]
listaM3=[]



def completar(x):
    if(x<10):
        dato="00"+str(x)

    elif (x<100):
        dato="0"+str(x)

    elif (x<1000):
        dato=str(x)
    return(dato)



def cargarArchivoM1():
    print("\t cargando archivos memoria 1..")

    #Condición: El archivo de datos debe estar en la ruta del programa
    listaM1.clear() # Eliminamos el contenido de la lista
    archivo = open("memoria1.txt", "r")
    lineas = archivo.readlines() # Subimos todas las líneas del archivo a lineas
    lista = list(lineas) # Convertimos las líneas en lista

    if(len(lista)!=0):
        for i in range(len(lista)):
            # Recorremos toda la lista
```

```
            listaM1.append([]) # Por ser primera fila se agrega la fila en blanco
            m1 =lista[i] # tomamos el valor de la lista y lo guardamos en codigo
            p=len(m1)
            print(p)
            if(p==24):
                listaM1[i].append(int(m1[0:3]))
                listaM1[i].append(int(m1[4:7]))
                listaM1[i].append(int(m1[8:11]))
                listaM1[i].append(int(m1[12:15]))
                listaM1[i].append(int(m1[16:19]))
                listaM1[i].append(int(m1[20:23]))
            else:
                listaM1[i].append(int(m1[0:2]))
                listaM1[i].append(int(m1[3:6]))
                listaM1[i].append(int(m1[7:10]))
                listaM1[i].append(int(m1[11:14]))
                listaM1[i].append(int(m1[15:18]))
                listaM1[i].append(int(m1[19:22]))

        print(listaM1)
        archivo.close()#3. Cerrar el archivo
        print("\t Terminamos de cargar archivo..")
    else:
        print("\t la agenda no tiene registros")

def cargarArchivoM2():
    print("\t cargando archivos memoria 2..")

    #Condición: El archivo de datos debe estar en la ruta del programa
    listaM2.clear() # Eliminamos el contenido de la lista
    archivo = open("memoria2.txt", "r")
    lineas = archivo.readlines() # Subimos todas las líneas del archivo a lineas
    lista = list(lineas) # Convertimos las líneas en lista

    if(len(lista)!=0):
        for i in range(len(lista)):
            # Recorremos toda la lista
            listaM2.append([]) # Por ser primera fila se agrega la fila en blanco
            m1 =lista[i] # tomamos el valor de la lista y lo guardamos en codigo
            p=len(m1)
            print(p)
            if(p==24):
                listaM2[i].append(int(m1[0:3]))
```

```
                listaM2[i].append(int(m1[4:7]))
                listaM2[i].append(int(m1[8:11]))
                listaM2[i].append(int(m1[12:15]))
                listaM2[i].append(int(m1[16:19]))
                listaM2[i].append(int(m1[20:23]))
            else:
                listaM2[i].append(int(m1[0:2]))
                listaM2[i].append(int(m1[3:6]))
                listaM2[i].append(int(m1[7:10]))
                listaM2[i].append(int(m1[11:14]))
                listaM2[i].append(int(m1[15:18]))
                listaM2[i].append(int(m1[19:22]))


        print(listaM2)
        archivo.close()#3. Cerrar el archivo
        print("\t Terminamos de cargar archivo..")
    else:
        print("\t la agenda no tiene registros")

def cargarArchivoM3():
    print("\t cargando archivos memoria 3..")

    #Condición: El archivo de datos debe estar en la ruta del programa
    listaM3.clear() # Eliminamos el contenido de la lista
    archivo = open("memoria3.txt", "r")
    lineas = archivo.readlines() # Subimos todas las líneas del archivo a lineas
    lista = list(lineas) # Convertimos las líneas en lista

    if(len(lista)!=0):
        for i in range(len(lista)):
            # Recorremos toda la lista
            listaM3.append([]) # Por ser primera fila se agrega la fila en blanco
            m1 =lista[i] # tomamos el valor de la lista y lo guardamos en codigo
            p=len(m1)
            print(p)
            if(p==24):
                listaM3[i].append(int(m1[0:3]))
                listaM3[i].append(int(m1[4:7]))
                listaM3[i].append(int(m1[8:11]))
                listaM3[i].append(int(m1[12:15]))
                listaM3[i].append(int(m1[16:19]))
                listaM3[i].append(int(m1[20:23]))
```

```
            else:
                listaM3[i].append(int(m1[0:2]))
                listaM3[i].append(int(m1[3:6]))
                listaM3[i].append(int(m1[7:10]))
                listaM3[i].append(int(m1[11:14]))
                listaM3[i].append(int(m1[15:18]))
                listaM3[i].append(int(m1[19:22]))

        print(listaM3)
        archivo.close()#3. Cerrar el archivo
        print("\t Terminamos de cargar archivo..")
    else:
        print("\t la agenda no tiene registros")




def modulo1():
    temp=slider1.get()
    temp2=slider2.get()
    temp3=slider3.get()
    temp4=slider4.get()
    temp5=slider5.get()
    temp6=slider6.get()
    cadena=completar(temp6)+","+completar(temp)+","+completar(temp2)+","+completa
r(temp3)+","+completar(temp4)+","+completar(temp5)+"z"
    print(cadena)
   #arduino.write(cadena.encode())

def memoria1 ():
    temp=slider1.get()
    temp2=slider2.get()
    temp3=slider3.get()
    temp4=slider4.get()
    temp5=slider5.get()
    temp6=slider6.get()

    numero=len(listaM1)
    listaM1.append([])
    listaM1[numero].append(temp6)
    listaM1[numero].append(temp)
    listaM1[numero].append(temp2)
    listaM1[numero].append(temp3)
```

```
    listaM1[numero].append(temp4)
    listaM1[numero].append(temp5)

    archivo=open('memoria1.txt','w')
    for dato in listaM1:
        archivo.write(str(dato[0]))
        archivo.write(',')
        archivo.write(str(dato[1]))
        archivo.write(',')
        archivo.write(str(dato[2]))
        archivo.write(',')
        archivo.write(str(dato[3]))
        archivo.write(',')
        archivo.write(str(dato[4]))
        archivo.write(',')
        archivo.write(str(dato[5]))
        archivo.write('\n')
    archivo.close


def memoria2():
    temp=slider1.get()
    temp2=slider2.get()
    temp3=slider3.get()
    temp4=slider4.get()
    temp5=slider5.get()
    temp6=slider6.get()

    numero=len(listaM1)
    listaM1.append([])
    listaM1[numero].append(temp6)
    listaM1[numero].append(temp)
    listaM1[numero].append(temp2)
    listaM1[numero].append(temp3)
    listaM1[numero].append(temp4)
    listaM1[numero].append(temp5)

    archivo=open('memoria2.txt','w')
    for dato in listaM1:
        archivo.write(str(dato[0]))
        archivo.write(',')
        archivo.write(str(dato[1]))
        archivo.write(',')
```

```
        archivo.write(str(dato[2]))
        archivo.write(',')
        archivo.write(str(dato[3]))
        archivo.write(',')
        archivo.write(str(dato[4]))
        archivo.write(',')
        archivo.write(str(dato[5]))
        archivo.write('\n')
    archivo.close


def memoria3():
    temp=slider1.get()
    temp2=slider2.get()
    temp3=slider3.get()
    temp4=slider4.get()
    temp5=slider5.get()
    temp6=slider6.get()

    numero=len(listaM1)
    listaM1.append([])
    listaM1[numero].append(temp6)
    listaM1[numero].append(temp)
    listaM1[numero].append(temp2)
    listaM1[numero].append(temp3)
    listaM1[numero].append(temp4)
    listaM1[numero].append(temp5)

    archivo=open('memoria3.txt','w')
    for dato in listaM1:
        archivo.write(str(dato[0]))
        archivo.write(',')
        archivo.write(str(dato[1]))
        archivo.write(',')
        archivo.write(str(dato[2]))
        archivo.write(',')
        archivo.write(str(dato[3]))
        archivo.write(',')
        archivo.write(str(dato[4]))
        archivo.write(',')
        archivo.write(str(dato[5]))
        archivo.write('\n')
    archivo.close
```

```
def secuencia(x):
    for dato in x:
            cadena=cadena=completar(dato[0])+","+completar(dato[1])+","+completar
(dato[2])+","+completar(dato[3])+","+completar(dato[4])+","+completar(dato[5])+"z
"
            #arduino.write(cadena.encode())
            time.sleep(1.6)



def desplazamiento():
    for r in range (2):
        secuencia(listaM1)
        print(str(r))
    print("primer movimiento")
    secuencia(listaM2)
    print("segundo movimiento")
    for r in range (2):
        secuencia(listaM2)
        print(str(r))
    print("tercer movimiento")
    secuencia(listaM2)
    for r in range (2):
        secuencia(listaM3)
        print(str(r))
    print("cuarto movimiento")
    secuencia(listaM2)
    print("quinto movimiento")
    for r in range (2):
        secuencia(listaM1)
        print(str(r))
def movimiento1():
    for r in range (10):
        print(str(r))

        for dato in listaM1:

            cadena=completar(dato[0])+","+completar(dato[1])+","+completar(dato[2
])+","+completar(dato[3])+","+completar(dato[4])+","+completar(dato[5])+"z"
            print(cadena)
            #arduino.write(cadena.encode())
            time.sleep(1.6)
```

```
    print("Terminé")
def movimiento2():
    for dato in listaM2:
        cadena=completar(dato[0])+","+completar(dato[1])+","+completar(dato[2])+"
,"+completar(dato[3])+","+completar(dato[4])+","+completar(dato[5])+"#"
        print(cadena)
        #arduino.write(cadena.encode())
        time.sleep(2)

    print("Terminé")
def movimiento3():
    for dato in listaM3:
        cadena=cadena=completar(dato[0])+","+completar(dato[1])+","+completar(dat
o[2])+","+completar(dato[3])+","+completar(dato[4])+","+completar(dato[5])+"z"
        print(cadena)
        arduino.write(cadena.encode())
        time.sleep(2)


ventana=Tk()

ventana.geometry("600x600+100+100")
ventana.title("CONTROL ROBOT EMERGE")

imagenl=PhotoImage(file="img.gif")
imagenl=imagenl.subsample(8,8)
lblImagen=Label(image=imagenl).place(x=400,y=0)

imagen2=PhotoImage(file="logo.gif")
imagen2=imagen2.subsample(2,2)
lblImagen2=Label(image=imagen2).place(x=0,y=420)



label = Label(ventana, text="ROBOT EMERGE")
label.pack(anchor=CENTER)
label.config(fg="blue",    # Foregroun
             font=("Verdana",20))



slider1=IntVar()
```

```
w = Scale(ventana,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider1)
w.set(512)
w.place(x=190,y=60)


slider2=IntVar()
w1 = Scale(ventana,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider2)
w1.set(512)
w1.place(x=190,y=120)

slider3=IntVar()
w3 = Scale(ventana,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider3)
w3.set(512)
w3.place(x=190,y=180)

slider4=IntVar()
w4 = Scale(ventana,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider4)
w4.set(512)
w4.place(x=190,y=240)

slider5=IntVar()
w5 = Scale(ventana,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider5)
w5.set(512)
w5.place(x=190,y=300)

slider6=IntVar()
w6 = Scale(ventana,from_=30, to=200,
length=230,tickinterval=30,orient=HORIZONTAL,variable=slider6)
w6.set(90)
w6.place(x=190,y=360)

lb1=Label(ventana,text="MODULO 1", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=74)
lb2=Label(ventana,text="MODULO 2", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=130)
lb3=Label(ventana,text="MODULO 3", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=193)
```

```
lb4=Label(ventana,text="MODULO 4", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=253)
lb5=Label(ventana,text="MODULO 5", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=313)
lb6=Label(ventana,text="VELOCIDAD", fg ='navy',font=('Segoe
Script',15)).place(x=40,y=380)


bt1=Button(ventana,text=" MOVER ",command=modulo1,font=("Agency
FB",12)).place(x=150,y=460)
bt2=Button(ventana,text=" memoria 1  ",command=memoria1,font=("Agency
FB",12)).place(x=500,y=160)
bt3=Button(ventana,text=" memoria 2 ",command=memoria2,font=("Agency
FB",12)).place(x=500,y=200)
bt4=Button(ventana,text=" memoria 3 ",command=memoria3,font=("Agency
FB",12)).place(x=500,y=240)

bt5=Button(ventana,text="movimento M1 ",command=movimiento1,font=("Agency
FB",12)).place(x=250,y=430)
bt6=Button(ventana,text="movimento M2",command=movimiento2,font=("Agency
FB",12)).place(x=250,y=470)
bt7=Button(ventana,text="movimento M3",command=movimiento3,font=("Agency
FB",12)).place(x=250,y=510)
bt11=Button(ventana,text="desplazamiento",command=desplazamiento,font=("Agency
FB",12)).place(x=460,y=460)

bt8=Button(ventana,text="CARGAR M1",command=cargarArchivoM1,font=("Agency
FB",12)).place(x=360,y=430)
bt9=Button(ventana,text="CARGAR M2",command=cargarArchivoM2,font=("Agency
FB",12)).place(x=360,y=470)
bt10=Button(ventana,text="CARGAR M3",command=cargarArchivoM3,font=("Agency
FB",12)).place(x=360,y=510)
ventana.mainloop()
```

10.3. Anexo 3

Algoritmo de entrenamiento de las redes neuronales

```
import numpy as np
import tensorflow as tf
from tensorflow.python.keras.layers import Dense
```

```
from tensorflow.python.keras import Sequential
from keras.utils import to_categorical
from keras.models import load_model
from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix

class Red:
    def __init__(self,x,y):
        self.Entrada=x
        self.Objetivo=y

    def _mean_relative_error(normalizer):
        def mean_relative_error(y_pred, y_true):
            return tf.math.divide_no_nan(tf.abs(y_true - y_pred), normalizer)
        return mean_relative_error


    def Entrenar(self):
        self.Modelo = Sequential()
        # Defina la cantidad de capas ocultas y las dimensiones de la entrada
        # En este caso es la longitud de la imagen vectorizada
        # Configure la capa de salida

        self.Modelo.add(Dense(len(self.Objetivo[0]), activation='relu'))#''sigmoi

        self.Modelo.add(Dense(35, activation='relu'))
        self.Modelo.add(Dense(160, activation='relu'))
        self.Modelo.add(Dense(160, activation='sigmoid'))
        self.Modelo.add(Dense(260, activation='relu'))
        self.Modelo.add(Dense(260, activation='softmax'))
        self.Modelo.add(Dense(260, activation='tanh'))
        self.Modelo.add(Dense(260, activation='softmax'))
        self.Modelo.add(Dense(260, activation='tanh'))
        self.Modelo.add(Dense(260, activation='sigmoid'))
        self.Modelo.add(Dense(260, activation='relu'))
        self.Modelo.add(Dense(260, activation='softmax'))
        self.Modelo.add(Dense(260, activation='relu'))
        self.Modelo.add(Dense(260, activation='softmax'))

        self.Modelo.add(Dense(480, activation='tanh'))
        self.Modelo.add(Dense(480, activation='sigmoid'))
        self.Modelo.add(Dense(480, activation='relu'))
        self.Modelo.add(Dense(480, activation='softmax'))
```

```
        self.Modelo.add(Dense(480, activation='relu'))
        self.Modelo.add(Dense(480, activation='softmax'))
        self.Modelo.add(Dense(480, activation='softmax'))
        self.Modelo.add(Dense(480, activation='softmax'))

        self.Modelo.add(Dense(840, activation='tanh'))
        self.Modelo.add(Dense(840, activation='sigmoid'))
        self.Modelo.add(Dense(840, activation='relu'))
        self.Modelo.add(Dense(840, activation='softmax'))
        self.Modelo.add(Dense(840, activation='relu'))
        self.Modelo.add(Dense(840, activation='softmax'))

        self.Modelo.add(Dense(800, activation='tanh'))
        self.Modelo.add(Dense(800, activation='sigmoid'))
        self.Modelo.add(Dense(800, activation='relu'))
        self.Modelo.add(Dense(800, activation='softmax'))
        self.Modelo.add(Dense(700, activation='relu'))
        self.Modelo.add(Dense(700, activation='softmax'))



        self.Modelo.add(Dense(360, activation='softmax'))
        self.Modelo.add(Dense(360, activation='softmax'))
        self.Modelo.add(Dense(360, activation='tanh'))
        self.Modelo.add(Dense(360, activation='sigmoid'))
        self.Modelo.add(Dense(500, activation='relu'))
        self.Modelo.add(Dense(500, activation='softmax'))
        self.Modelo.add(Dense(500, activation='relu'))

        self.Modelo.add(Dense(500, activation='linear'))
        self.Modelo.add(Dense(500, activation='linear'))
        self.Modelo.add(Dense(500, activation='linear'))

        self.Modelo.add(Dense(400, activation='linear'))
        self.Modelo.add(Dense(400, activation='linear'))
        self.Modelo.add(Dense(560, activation='linear'))
        self.Modelo.add(Dense(660, activation='linear'))
        self.Modelo.add(Dense(360, activation='linear'))
        self.Modelo.add(Dense(360, activation='linear'))

        self.Modelo.add(Dense(360, activation='softmax'))
        self.Modelo.add(Dense(360, activation='softmax'))
        self.Modelo.add(Dense(360, activation='softmax'))
```

```
        self.Modelo.add(Dense(360, activation='relu'))
        self.Modelo.add(Dense(360, activation='relu'))
        self.Modelo.add(Dense(35, activation='relu'))




        #self.Modelo.add(Dense(layers.Dense(32, activation='relu'))#''sigmoid


        #Configure los parámetros de entrenamiento del modelo
        #self.Modelo.compile(loss="poisson", #poisson
        #optimizer="nadam",
        #metrics=['accuracy'])


        self.Modelo.compile(loss='poisson',
              optimizer='nadam',              metrics=['accuracy'])


        # Entrenar el modelo
        history =self.Modelo.fit(self.Entrada, self.Objetivo, batch_size=1000,
epochs=7500, validation_data=(self.Entrada, self.Objetivo))
        # Guardar RNA en una carpeta
        #self.modelo.save('path_to_my_model.h5')
        loss, acc = self.Modelo.evaluate(self.Entrada, self.Objetivo)
        print('Precisión:', acc)
        print (self.Modelo.predict(self.Entrada).round())

        self.Modelo.evaluate(self.Entrada, self.Objetivo)

        score=self.Modelo.evaluate(self.Entrada,
self.Objetivo,  batch_size=1000,verbose=True )
        print("val score: " ,score[0])
        print("val accurary: " ,score[1])

        plt.plot(history.history['accuracy'])
        plt.plot(history.history['val_accuracy'])
        plt.title('model accuracy')
        plt.ylabel('accuracy')
        plt.xlabel('epoch')
        plt.legend(['train', 'val'], loc='upper left')
        plt.show()
```

```
        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        plt.title('model Loss')
        plt.ylabel('Loss')
        plt.xlabel('epoch')
        plt.legend(['train', 'val'], loc='upper left')
        plt.show()

        self.Modelo.save('rna_b00');
        #z= np.array([[0,0]])
        #t=self.Modelo.predict(z)
        #print(t)



    def Validar(self):
        # Importar RNA
        self.Modelo = self.Modelo.load_model('cadena00')
        # Evaluar el modelo
        loss, acc = self.Modelo.evaluate(self.Entrada, self.Objetivo)
        print('Precisión:', acc)
        print (self.Modelo.predict(self.Entrada).round())

    def prediccion(self,o):
        print("todos")
        z= np.array([[o]])
        t=self.Modelo.predict(z)
        print(t)
        y_pred =self.Modelo.predict(z)

# Calcula la matriz de confusión
        y_pred_classes = np.round(y_pred)
        cm = confusion_matrix(self.Objetivo, y_pred)
        print(cm)


X= np.array([[0]])
print(X)
global cadena00, cadena10, cadena01, cadena11,ele00,ele01,
ele10,ele11,bloque00,bloque01,bloque10,bloque11
cadena00=[]
cadena01=[]
```

```
cadena10=[]
cadena11=[]
ele00=[]
ele01=[]
ele10=[]
ele11=[]
bloque00=[]
bloque01=[]
bloque10=[]
bloque11=[]

def cargarArchivo(x):
    print("\t cargando archivos ...  "+x+"  ...")
    listaM2=[]
    #Condición: El archivo de datos debe estar en la ruta del programa
    listaM2.clear() # Eliminamos el contenido de la lista
    archivo = open(x, "r")
    lineas = archivo.readlines() # Subimos todas las líneas del archivo a lineas
    lista = list(lineas) # Convertimos las líneas en lista

    if(len(lista)!=0):
        for i in range(len(lista)):
            # Recorremos toda la lista
            listaM2.append([]) # Por ser primera fila se agrega la fila en blanco
            m1 =lista[i] # tomamos el valor de la lista y lo guardamos en codigo
            listaM2[i].append(int(m1[0:3]))
            listaM2[i].append(int(m1[4:7]))
            listaM2[i].append(int(m1[8:11]))
            listaM2[i].append(int(m1[12:15]))
            listaM2[i].append(int(m1[16:19]))
        archivo.close()#3. Cerrar el archivo
        print("\t Terminamos de cargar archivo..")
        return(listaM2)

    else:
        print("\t la agenda no tiene registros")

cadena00=cargarArchivo("cad00.txt")
#print(cadena00)
cadena01=cargarArchivo("cad01.txt")
#print(cadena01)
cadena10=cargarArchivo("cad10.txt")
#print(cadena10)
```

```
cadena11=cargarArchivo("cad11.txt")
#print(cadena11)
ele00=cargarArchivo("ele00.txt")
#print(ele00)
ele01=cargarArchivo("ele01.txt")
#print(ele01)
ele10=cargarArchivo("ele10.txt")
#print(ele10)
ele11=cargarArchivo("ele11.txt")
#print(ele11)
bloque00=cargarArchivo("blok00.txt")
#print(bloque00)
bloque01=cargarArchivo("blok01.txt")
#print(bloque01)
bloque10=cargarArchivo("blok10.txt")
#print(bloque10)
bloque11=cargarArchivo("blok11.txt")
#print(bloque11)

y= np.array([[bloque00]])
#y=
np.array([[512,512,512,512,512],[312,339,512,712,512],[512,312,512,339,712],[512,
312,512,339,712],[512,512,512,512,312]])
print(y)
print(len(y))
print(len(X))
print(y.reshape(1,-1))
y=y.reshape(1,-1)
#y=y/np.linalg.norm(y)
print(y)

if __name__ == "__main__":
    # Crear los arreglos de entrada y de salida

    # Convertir los valores de la matriz de rojos en enteros
    #X = X.reshape(5,-1)
    # Convetir los valores de salida en variables categoricas
    #y = np.array(y, dtype = "uint8")

    # Vectorizar los valores para entrenar la red
    # Convetir los valores de salida en variables categoricas
    # Crear la red
    X = X.reshape(1,-1)
```

```
    RNA=Red(X,y)
    RNA.Entrenar()
    print(y)
    RNA.Validar()

    t= RNA.prediccion(0)
```

10.4. Anexo 4

Algoritmo de entrenamiento de las redes neuronales para el control de movimiento del robot EMERGE.

```
import tensorflow as tf
from tensorflow import keras
import numpy as np

# Definir los datos de entrada y salida
X = np.array([0])
y= np.array([
[
511,512,512,512,511,
550,512,512,512,493,
550,512,512,512,462,
550,306,512,755,462,
550,306,677,755,462,
550,511,677,511,462,
511,511,511,511,462,
511,511,511,511,511,
511,512,512,512,511,
550,512,512,512,493,
550,512,512,512,462,
550,306,512,755,462,
550,306,677,755,462,
550,511,677,511,462,
511,511,511,511,462,
511,511,511,511,511

]
          ])

# Definir la estructura de la red neuronal
model = keras.Sequential([
```

```
    keras.layers.Dense(units=80, input_shape=[1]),
    keras.layers.Dense(units=80),
    keras.layers.Dense(units=80)
])

# Compilar la red neuronal
model.compile(optimizer='nadam', loss='mean_squared_error')

# Entrenar la red neuronal
history=model.fit(X, y, epochs=1000)

# Predecir con la red neuronal
prediction = model.predict(X)
print(prediction)

y_pred = model.predict(X)

t=X = np.array([0])
y_pred = model.predict(t)
print(y_pred)
print(y[0])

from google.colab import drive
drive.mount('/content/gdrive')

model.save("/content/gdrive/My Drive/cadena10v2.h5")
```

10.5. Anexo 5

Algoritmo de entrenamiento red neuronal sensores de detección de obstáculos.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense
import numpy as np

# Datos de entrada y salida
X = np.array([100, 101, 110, 111, 200, 201, 210, 211, 300, 301, 310, 311])
y = np.array(['cadena00', 'cadena01', 'cadena10', 'cadena11', 'ele00', 'ele01',
'ele10', 'ele11', 'bloque00', 'bloque01', 'bloque10', 'bloque11'])

# Convertir las etiquetas a valores numéricos
unique_labels = np.unique(y)
label_map = {label: i for i, label in enumerate(unique_labels)}
```

```
y = np.array([label_map[label] for label in y])

# Crear modelo de red neuronal
model = keras.Sequential([
    Dense(20, activation='relu', input_shape=(1,)),
    Dense(600, activation='sigmoid'),
    Dense(600, activation='linear'),
    Dense(12, activation='softmax')
])

# Compilar el modelo
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Entrenar el modelo
model.fit(X, y, epochs=1600, batch_size=30)

# Hacer predicciones con el modelo
t=np.array([100])

predictions = model.predict(X)

# Convertir las predicciones a etiquetas
predicted_labels = [unique_labels[np.argmax(pred)] for pred in predictions]
print(predicted_labels)


predictions = model.predict(X)

# Convertir las predicciones a etiquetas
predicted_labels = [unique_labels[np.argmax(pred)] for pred in predictions]
print(predicted_labels)
t=np.array([200])

predictions = model.predict(t)

# Convertir las predicciones a etiquetas
predicted_labels = [unique_labels[np.argmax(pred)] for pred in predictions]
print(predicted_labels)

from google.colab import drive
drive.mount('/content/gdrive')
```

```
model.save("/content/gdrive/My Drive/orquestador.h5")
```

10.6. Anexo 6

```
from tensorflow.keras.models import load_model
import numpy as np
import tkinter as tk
from tkinter import *
import time
import serial
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
arduino = serial.Serial(
    port='/dev/serial0',  # Cambiar por el puerto que estés utilizando
    baudrate=9600,
    timeout=1
)

obstaculo=30
sensores=[0,0]
salida=[0,0]
morfologia=0
velocidad=150

y = np.array(['cadena00', 'cadena01', 'cadena10', 'cadena11', 'ele00', 'ele01',
'ele10', 'ele11', 'bloque00', 'bloque01', 'bloque10', 'bloque11'])
unique_labels = np.unique(y)
label_map = {label: i for i, label in enumerate(unique_labels)}
y = np.array([label_map[label] for label in y])

def completar(x):
    if(x<10):
        dato="00"+str(x)
    elif (x<100):
        dato="0"+str(x)
    elif (x<1000):
        dato=str(x)
    return(dato)

def distancia():
    global sensores, salida
    vec=['','','','','','','','','','']
```

```
    salida=[0,0]
    rec=0
    print("sensando...")
    cadena=''
    arduino.write("S".encode())
    while True:

        char=arduino.read()
        char=char.decode('ascii').strip()
        if filtro(char):
            print(char)
            vec[rec]=char
            rec=rec+1
        if char=='#':
            print(vec)
            extraer(vec)
            print(sensores)
            rec=0
            break


def filtro(x):
    r=False
    if x in {'0','1','2','3','4','5','6','7','8','9',',','#'}:
        r=True
    return(r)

def  extraer(t):
    global obstaculo, sensores, salida
    salida=[0,0]
    if t[2]==',':
        sensores[0]=numero(t[0])*10 + numero(t[1])
        if t[5]=='#':
            sensores[1]=numero(t[3])*10 +numero(t[4])
        elif t[6]=='#':
            sensores[1]=numero(t[3])*100 +numero(t[4])*10+numero(t[5])

    elif t[3]==',':
       sensores[0]=numero(t[0])*100 + numero(t[1])*10 + numero(t[2])
       if t[6]=='#':
           sensores[1]=numero(t[4])*10 +numero(t[5])
       elif t[7]=='#':
           sensores[1]=numero(t[4])*100 +numero(t[5])*10+numero(t[6])
```

```
    print(sensores)
    if sensores[0]< obstaculo:
        salida[0]=1
    if sensores[1]< obstaculo:
        salida[1]=1
    #print(salida)

def numero(x):
    r=0
    if x in {'0','1','2','3','4','5','6','7','8','9'}:
        r=int(x)
    return(r)

def movimiento():
    #
    temp=slider1.get()
    temp2=slider2.get()
    temp3=slider3.get()
    temp4=slider4.get()
    temp5=slider5.get()
    temp6=slider6.get()
    cadena="Z"+completar(temp6)+","+completar(temp)+","+completar(temp2)+","+comp
letar(temp3)+","+completar(temp4)+","+completar(temp5)+"#"
    #cadena='A'
    print(cadena)
    arduino.write(cadena.encode())


def recorrido():
#bucle while
    try:
        while True:
            global orquestador, morfologia, sensores, velocidad, salida

            distancia()
            t = morfologia+(salida[0]*10)+ salida[1]
            print(t)
            t=np.array([t])

            predictions = orquestador.predict(t)

            # Convertir las predicciones a etiquetas
```

```
            predicted_labels = [unique_labels[np.argmax(pred)] for pred in
predictions]
            #print(predicted_labels)
            charlie = predicted_labels[0]
            print(charlie)

            valor = np.array([0])
            prediccion = predecir(charlie, valor)

            x = prediccion
            l = len(x[0])
            print (l)
            secciones = x.reshape(-1, 5)
            print(secciones)
            for i in range(secciones.shape[0]):
                cadena = ','.join(str(int(x)) for x in secciones[i])
                cadena = "Z"+ str(velocidad)+"," + cadena + "#"
                print(cadena)
                arduino.write(cadena.encode())
                time.sleep(1.5)
            #time.sleep(0.6)


    except KeyboardInterrupt:
        print('interrupted!')



def cargar_imagen():
    global morfologia, velocidad , obstaculo
    imagen = tk.PhotoImage(file="cadena.png")
    imagen=imagen.subsample(2,2)
    imagen_label.configure(image=imagen)
    imagen_label.image = imagen
    morfologia=100
    velocidad=200
    obstaculo=30

def cargar_imagen1():
    global morfologia, velocidad, obstaculo
    imagen = tk.PhotoImage(file="ele.png")
    imagen=imagen.subsample(2,2)
    imagen_label.configure(image=imagen)
```

```
    imagen_label.image = imagen
    morfologia=200
    velocidad=150
    obstaculo=35

def cargar_imagen2():
    global morfologia, velocidad, obstaculo
    imagen = tk.PhotoImage(file="bloque.png")
    imagen=imagen.subsample(3,3)
    imagen_label.configure(image=imagen)
    imagen_label.image = imagen
    morfologia=300
    velocidad=150
    obstaculo=35


def cague_redes():

    global modelos, orquestador
    print ("realizando el cargue...")
    modelo1 = load_model('cadena00v2.h5')
    print ("RNA 1 cargada con exito..")
    modelo2 = load_model('cadena01v2.h5')
    print ("RNA 2 cargada con exito..")
    modelo3 = load_model('cadena10v2.h5')
    print ("RNA 3 cargada con exito..")
    modelo4 = load_model('cadena11v2.h5')
    print ("RNA 4 cargada con exito..")

    modelo5 = load_model('ele00v2.h5')
    print ("RNA 5 cargada con exito..")
    modelo6 = load_model('ele01v2.h5')
    print ("RNA 6 cargada con exito..")
    modelo7 = load_model('ele10v2.h5')
    print ("RNA 7 cargada con exito..")
    modelo8 = load_model('ele11v2.h5')
    print ("RNA 8 cargada con exito..")

    modelo9 = load_model('bloque00v2.h5')
    print ("RNA 9 cargada con exito..")
    modelo10 = load_model('bloque01v2.h5')
    print ("RNA 10 cargada con exito..")
    modelo11 = load_model('bloque10v2.h5')
```

```
    print ("RNA 11 cargada con exito..")
    modelo12 = load_model('bloque11v2.h5')
    print ("RNA 12 cargada con exito..")

    orquestador = load_model('nosgraduamosv5.h5')
    print ("RNA ORQUESTADOR cargado con exito..")


    modelos = {
    'cadena00': modelo1,
    'cadena01': modelo2,
    'cadena10': modelo3,
    'cadena11': modelo4,

    'ele00': modelo5,
    'ele01': modelo6,
    'ele10': modelo7,
    'ele11': modelo8,

    'bloque00': modelo9,
    'bloque01': modelo10,
    'bloque10': modelo11,
    'bloque11': modelo12

    }
    print ("memorias cargadas con exito...")

def predecir(etiqueta, valor):
    global modelos
    modelo = modelos[etiqueta]
    return modelo.predict(valor)




def abrir_ventana():

    global slider1,slider2,slider3,slider4,slider5,slider6
    ventana1 = tk.Toplevel()
    ventana1.title("Nueva ventana")
    ventana1.geometry("500x470")
    slider1=IntVar()
```

```
    w = Scale(ventana1,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider1)
    w.set(512)
    w.place(x=190,y=30)


    slider2=IntVar()
    w1 = Scale(ventana1,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider2)
    w1.set(512)
    w1.place(x=190,y=90)

    slider3=IntVar()
    w3 = Scale(ventana1,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider3)
    w3.set(512)
    w3.place(x=190,y=150)

    slider4=IntVar()
    w4 = Scale(ventana1,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider4)
    w4.set(512)
    w4.place(x=190,y=210)

    slider5=IntVar()
    w5 = Scale(ventana1,from_=200, to=900,
length=230,tickinterval=200,orient=HORIZONTAL,variable=slider5)
    w5.set(512)
    w5.place(x=190,y=270)

    slider6=IntVar()
    w6 = Scale(ventana1,from_=30, to=200,
length=230,tickinterval=30,orient=HORIZONTAL,variable=slider6)
    w6.set(90)
    w6.place(x=190,y=330)

    lb1=Label(ventana1,text="MODULO 1", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=44)
    lb2=Label(ventana1,text="MODULO 2", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=100)
    lb3=Label(ventana1,text="MODULO 3", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=163)
```

```
    lb4=Label(ventana1,text="MODULO 4", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=223)
    lb5=Label(ventana1,text="MODULO 5", fg ='navy',font=('Segoe
Script',15)).place(x=50,y=283)
    lb6=Label(ventana1,text="VELOCIDAD", fg ='navy',font=('Segoe
Script',15)).place(x=40,y=350)
    bt1=Button(ventana1,text=" MOVER ",command=movimiento,font=("Agency
FB",12)).place(x=220,y=410)



ventana = tk.Tk()
ventana.geometry("600x600+100+100")
ventana.title("CONTROL ROBOT EMERGE")


imagen2=PhotoImage(file="logo.gif")
imagen2=imagen2.subsample(2,2)
lblImagen2=Label(image=imagen2)
lblImagen2.place(x=0,y=450)


label = Label(ventana, text="ROBOT EMERGE")
label.pack(anchor=CENTER)
label.config(fg="black",    # Foregroun
             font=("Verdana",20))

imagen_label = tk.Label(ventana)
imagen_label.pack(pady=10)

boton = tk.Button(ventana, text="TEST", command=abrir_ventana).place(x=200,y=340)
boton = tk.Button(ventana, text="INICIAR RECORRIDO",
command=recorrido).place(x=200,y=440)
boton = tk.Button(ventana, text="CARGAR REDES",
command=cague_redes).place(x=200,y=480)
boton = tk.Button(ventana, text="CADENA",
command=cargar_imagen).place(x=150,y=400)
boton = tk.Button(ventana, text="ELE", command=cargar_imagen1).place(x=250,y=400)
boton = tk.Button(ventana, text="BLOQUE",
command=cargar_imagen2).place(x=350,y=400)


etiqueta1 = tk.Label(ventana, text="Natalia Rodriguez Gonzalez", font=("Arial",
10))
```

```
etiqueta1.place(x=350,y=520)


etiqueta2 = tk.Label(ventana, text="Yeyffer Arbey Montes Fonseca", font=("Arial",
10))
etiqueta2.place(x=350,y=540)

etiqueta2 = tk.Label(ventana, text="Ingenieria en control y automatización",
font=("Arial", 10))
etiqueta2.place(x=350,y=560)


ventana.mainloop()
```